Diego Heitzmann Quintale

Uma abordagem orientada a modelos para desenvolvimento de sistemas ERP de varejo na Web utilizando características funcionais de usabilidade

Sorocaba
2015

Universidade Federal de São Carlos
Programa de Pós-Graduação em Ciência da Computação (PPGCCS)

Diego Heitzmann Quintale

Uma abordagem orientada a modelos para desenvolvimento de sistemas ERP de varejo na Web utilizando características funcionais de usabilidade

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCCS) da Universidade Federal de São Carlos como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação área de concentração: Engenharia de Software e Gestão do Conhecimento.

Orientadora: Profa. Dra. Luciana Aparecida Martinez Zaina

Este exemplar corresponde à versão final da dissertação defendida pelo aluno, e orientada pela Profa. Dra. Luciana Aparecida Martinez Zaina

---

Sorocaba
2015

Quintale, Diego Heitzmann.
Q7a Uma abordagem orientada a modelos para desenvolvimento de sistemas ERP de varejo na Web utilizando características funcionais de usabilidade / Diego Heitzmann Quintale. – – 2015.
149 f. : 30 cm.

Dissertação (mestrado)-Universidade Federal de São Carlos, *Campus* Sorocaba, Sorocaba, 2015.
Orientador: Luciana Aparecida Martinez Zaina.
Banca examinadora: Renata Pontin de Mattos Fortes, Alexandre Alvaro.
Bibliografia

1. Engenharia de software. 2. Sites da Web - Desenvolvimento. I. Orientador. II. Sorocaba-Universidade Federal de São Carlos. III . Título.

CDD 005.12

Ficha catalográfica elaborada pela Biblioteca do *Campus* de Sorocaba.

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências em Gestão e Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Diego Heitzmann Quintale, realizada em 27/08/2015:

Profa. Dra. Luciana Aparecida Martinez Zaina
UFSCar

Profa. Dra. Renata Pontin de Mattos Fortes
USP

Prof. Dr. Alexandre Alvaro
UFSCar

Aos meus pais.

Agradecimentos

Agradeço,

Aos meus pais, Maria do Carmo e Orlando, pelo amor incondicional e incansável apoio.

A minha família e amigos pelos momentos, conselhos e orações.

A Profa. Dra. Luciana Zaina pelos cinco anos de convivência, de oportunidades e ensinamentos.

A Betalabs pelo suporte e tempo cedido.

Aos professores da UFSCar pelas experiências e paciência desprendidas ao longo dos anos.

A UFSCar e seus colaboradores pela oportunidade e estrutura oferecidas durante os anos de estudo.

A Deus pelas oportunidades e ajuda concedidas e por colocar todas essas pessoas no meu caminho.

A todos que, de alguma forma, contribuíram com o meu progresso.

Conheça todas as teorias, domine todas as técnicas, mas ao tocar uma alma humana, seja apenas outra alma humana.

Carl Jung

Resumo

Os ERPs são sistemas complexos que manipulam grande volume de dados e a ligação entre esses dados disponibilizando aos seus usuários diferentes informações. Normalmente, os desenvolvedores deste tipo de aplicação dedicam seus esforços na garantia de corretude das funcionalidades e consistência das informações; assim questões de usabilidade tornam-se secundárias. Assim é amplamente reconhecido que os sistemas ERP possuem interfaces complexas que afetam negativamente a capacidade de utilização destes sistemas. Entretanto, observa-se que certas regras que auxiliam na melhoria da interface podem ser automatizadas, e é possível criar transformadores automatizados para auxiliar no desenvolvimento destas aplicações. Esta dissertação apresenta uma abordagem dirigida a modelos para sistemas ERP de varejo com características funcionais de usabilidade (características funcionais com alto impacto na usabilidade), denominado FUF. Este estudo consistiu em duas etapas. Na primeira foi feito o levantamento bibliográfico e pesquisas de sistemas ERP de varejo de mercado. A segunda etapa consistiu no desenvolvimento de um metamodelo para sistemas ERP de varejo com características funcionais de usabilidade, denominado MetaUsaERPWeb. Também foi realizado o desenvolvimento de um transformador M2C para o MetaUsaERPWeb com o PHP como plataforma alvo. Por fim, na última etapa, a validação da proposta foi executada em um estudo de caso, com duas análises: primeiro realizou-se um estudo experimental para validar a abordagem proposta em relação a abordagem tradicional. Os resultados apontaram que os grupos que utilizaram o MetaUsaERPWeb foram mais eficientes, produziram mais linhas de código e aplicações de melhor qualidade. A segunda análise comparou as aplicações desenvolvidas a partir da abordagem tradicional e da utilização do MetaUsaERPWeb (primeira análise); os resultados mostraram que as aplicações desenvolvidas a partir do MetaUsaERPWeb violaram menos heurísticas de usabilidade em 60% dos casos.

Palavras-chave: ERP. Desenvolvimento Dirigido a Modelos. Heurísticas de usabilidade. Características funcionais de usabilidade.

Abstract

ERPs are complex systems that handle large amounts of data and links among this data providing to its users different information. Usually, developers of this type of application dedicate their efforts to guarantee correctness of the functionality and information consistency; so usability issues become secondary. It is widely recognized that ERP systems own complex interfaces that negatively affect the ability to use these systems. However, it is observed that certain rules that assist to improve interfaces may be automated, thus it is possible to create automated processes to help the development of such applications. This work presents a model driven approach for retail ERP systems with functional usability features (functional features with high impact on usability), called FUF. This study consisted in two stages. In the first one was made literature and market research of retail ERP systems. The second step was the development of a metamodel for retail ERP systems with functional usability features, called MetaUsaERPWeb, it was also carried out the development of a M2C transformer for MetaUsaERPWeb with PHP target platform. Finally, in the last step, the validation of the proposal was made in a case study, with two tests: the first conducted an experimental study to validate the proposed approach against the traditional approach. The results showed that the groups using MetaUsaERPWeb were more efficient, produce more lines of code and better applications. The second analysis compared the applications developed from the traditional approach and the use of MetaUsaERPWeb (first analysis); the results showed that applications built with MetaUsaERPWeb violated least usability heuristics in 60 % of cases.

Key-words: ERP. Model Driven Development. Usability heuristics. Functional usability features.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CA Afferent Coupling

CMS Content Management System

CRM Customer Relationship Management

CRUD Create, Read, Update, Delete

CSS Cascading Style Sheet

DSL Domain-Specific Language

DSM Domain-Specific Modeling

EA Engenharia de Aplicação

ED Engenharia de Domínio

ERP Enterprise Resource Planning

FUF Functional Usability Feature

HTML HyperText Markup Language

IHC Interação Humano-Computador

LCOM Lack of Cohesion of Methods

LOC Lines Of Code

M2C Model-to-Code

M2M Model-to-Model

MDA Model Driven Architecture

MDD Model Driven Development

MoU Mode of Use

MVC Model-View-Controller

NEE Número de Elementos Especificados

OMG Object Management Group

PC Personal Computer

PCP Planejamento e Controle de Produção

PHP PHP: Hypertext Preprocessor

PIM Plataform Independent Model

PSM Plataform Specific Model

REC Razão entre Especificação e Código

RFID Radio-Frequency IDentification

SPL Software Product Line

TAM Technology Acceptance Model

TCLE Termo de Consentimento Livre e Esclarecido

TI Tecnologia da Informação

UM Usability Mechanism

UML Unified Modeling Language

XMI XML Metadata Interchange

XML eXtensible Markup Language

Sumário

Capítulo 1

Introdução

ERPs (*Enterprise Resource Planning*) são sistemas usados por grandes, médias e pequenas empresas, sua adoção auxiliam o dia-a-dia da organização integrando processos e fornecendo informações de forma rápida e eficiente (WAGNER; MONK, 2008; QUIESCENTI et al., 2006). A aquisição desses sistemas é um alto investimento, pois a sua implementação é demorada, devido a sua complexidade e adequação do sistema à empresa, além dos testes que esse tipo de sistema exige.

O ERP era predominantemente desenvolvido para plataforma *desktop*. Atualmente, porém, tem migrado para a Web trazendo vantagens como melhoria na eficiência dos recursos de TI da empresa, facilidade na utilização e diminuição do custo de manutenção (JINPING; WENJIE, 2011). Além disso o sistema passa a estar disponível a qualquer hora e qualquer lugar também facilitando a sua utilização e acesso.

A utilização da plataforma Web para sistemas ERPs é possível graças ao crescimento rápido da Internet onde diversas tecnologias têm auxiliado neste processo. A ascensão da Web 2.0 é um exemplo desse crescimento, que encara a Web de uma forma diferente, trazendo ao usuário através do navegador a mesma experiência que os serviços *desktop* provêem.

Durante o desenvolvimento de um ERP os desenvolvedores focam a maior parte de seus esforços para garantir que o comportamento do sistema está correto e as infomacões se mantêm consistentes ao longo da utilização do sistema. Com isso, a interface e interação do usuário são colocadas em segundo plano já que não impacta diretamente no comportamento funcional do sistema. Embora a atividade de entrada de dados em um ERP tenha migrado para a automatização (leitor de código de barras, etiquetas RFID (Radio-Frequency IDentification), etc...) ainda é grande o uso de entrada manual principalmente nos sistemas de varejo de pequeno e médio porte (WAGNER; MONK, 2008).

A entrada de dados é um dos fatores que mais geram custos durante o uso do ERP e, por isso, a produtividade do usuário depende da eficiência dele mesmo (MILOSZ et al., 2013). Garantir a qualidade da interface gráfica com o usuário é um dos elementos chave para melhorar a eficiência e consistência dos dados do ERP e, além disso, melhorar a produtividade do usuário (HOLSAPPLE; WANG; WU, 2006; MILOSZ et al., 2013; PANACH; AQUINO; PASTOR, 2014).

1.1 Motivação e problema

Os ERPs são sistemas com funcionalidades que abrangem todas as áreas da organização, realizando as ligações e relações necessárias entre as informações. Essa é uma das causas do alto valor de um ERP e do seu alto tempo de desenvolvimento (MILOSZ et al., 2013). O termo ERP pode ser muito amplo visto que o sistema deve atender a empresa que o utilizará, como cada empresa tem uma demanda diferente cada ERP possui suas particularidades. Entretanto, apesar de suas disparidades, o sistema ERP possui características comuns e as variabilidades são construídas a partir delas.

É amplamente reconhecido que os sistemas ERP possuem interfaces complexas que afetam negativamente a capacidade de utilização desses sistemas (SINGH; WESSON, 2009). Designers de interação são desafiados a criarem uma interface de usuário para sistemas ERP, que atenda aos requisitos funcionais específicos e ao mesmo tempo seja aderente aos principais atributos de usabilidade, tais como capacidade de aprendizado, eficiência, memorização e satisfação. Para se manter a produtividade os sistemas ERP devem ser simples de entender e fácil de usar após período razoável de familiarização. Dessa forma a quantidade de complexidade percebida pelos usuários em sistemas ERP deve ser minimizada a fim de maximizar o desempenho das tarefas, a facilidade de uso e a satisfação do usuário (UFLACKER; BUSSE, 2007).

O *design* centrado no usuário surge como uma alternativa para minimizar as complexidades dos sistemas ERP, trazendo benefícios de aprendizado e interação com o sistema (UFLACKER; BUSSE, 2007). Além disso impactos de fatores de IHC em projetos de software ERP são uma diferenciação para alcançar vantagem competitiva nesse mercado (OZEN; BASOGLU, 2006). Apesar de questões relacionadas a interface serem conhecidas elas são deixadas em segundo plano pelos desenvolvedores já que seus esforços são direcionados a implementação das funcionalidades e garantir que a consistência das informações seja respeitada ao longo das operações do sistema.

1.2 Objetivos

1.2.1 Objetivo geral

O objetivo deste trabalho é propor uma abordagem dirigida a modelos de forma que, a partir de um metamodelo e seus respectivos transformadores de código, seja possível a modelagem e geração de código de módulos de um sistema ERP de varejo. O foco não é somente realizar a geração do código no âmbito funcional do domínio, mas também apresentar soluções no que tange ao aspecto de interação do usuário. Normalmente, a prevenção e recuperação de erros são elementos da interação que são deixados em segundo plano durante o desenvolvimento de sistemas altamente focados na visão funcional, ficando a cargo do desenvolvedor trabalhar com a prevenção e tratamento de erros. Contudo, grande parte destes elementos podem ser derivados das estruturas funcionais de entrada de dados que são especificadas durante a modelagem.

A proposta deste projeto busca relacionar estes elementos fundamentais de interação com os aspectos funcionais durante a modelagem. Os elementos de interação serão definidos e tratados com base nas heurísticas de Nielsen (NIELSEN, 1995a) e através das características funcionais de usabilidade (JURISTO; MORENO; SANCHEZ-SEGURA, 2007b).

Não está no escopo deste projeto a proposta de uma solução genérica para todas as necessidades e todas as frentes de sistemas ERP. Também não faz parte do objetivo deste trabalho a inclusão de todas melhorias de usabilidade possíveis.

1.2.2 Objetivos específicos e contribuições

A partir do objetivo geral são definidos objetivos específicos e contribuições:

- Levantar a bibliografia acerca de sistemas ERP, heurísticas de usabilidade e desenvolvimento dirigido a modelos;
- Coletar e identificar características funcionais e de usabilidade de sistemas ERP de mercado;
- Mapeamento entre características de usabilidade x heurísticas de usabilidade;
- Propor uma nova característica funcional de usabilidade (FUF) encontrada em sistemas ERPs analisados;
- Propor um metamodelo para sistemas ERP de varejo utilizando os dados coletados nas fases anteriores;
- Implementar um tranformador baseado no metamodelo proposto;
- Validar o metamodelo e o transformador proposto através da execução de um estudo de caso com desenvolvedores.

1.3 Metodologia e organização

A fim de atingir o objetivo geral e o específico este trabalho foi organizado a partir de técnicas de revisão bibliográfica e estudo de aplicações já existentes; na primeira etapa foi realizado o estudo bibliográfico bem como a coleta de informações de sistemas semelhantes de mercado. Na segunda etapa os esforços foram concentrados na definição e criação do metamodelo, seu transformador e validação da proposta. A Figura 1.1 apresenta a visão geral da metodologia utilizada.

A Etapa 1 consistiu em seis passos, onde no primeiro (a) foi definido o tema do trabalho, este passo embasou todos os passos posteriores até a conclusão das duas etapas. No passo seguinte foi feito o estudo bibliográfico (b) a fim de levantar o estado da arte a respeito dos principais assuntos deste trabalho: Desenvolvimento Dirigido a Modelos, sistemas ERP e princípios de usabilidade. O terceiro passo consistiu em um levantamento de funcionalidades de alguns sistemas ERP de mercado no domínio de varejo (c); já o próximo passo executou o levantamento das características de usabilidade destes sistemas (d). No passo (e) as análises dos dois passos anteriores foram compilados a fim de serem usados na etapa posterior. Por fim, o último passo (f) foi referente ao estudo das ferramentas EMF e JET, que foram utilizadas para elaboração do metamodelo e seus transformadores.

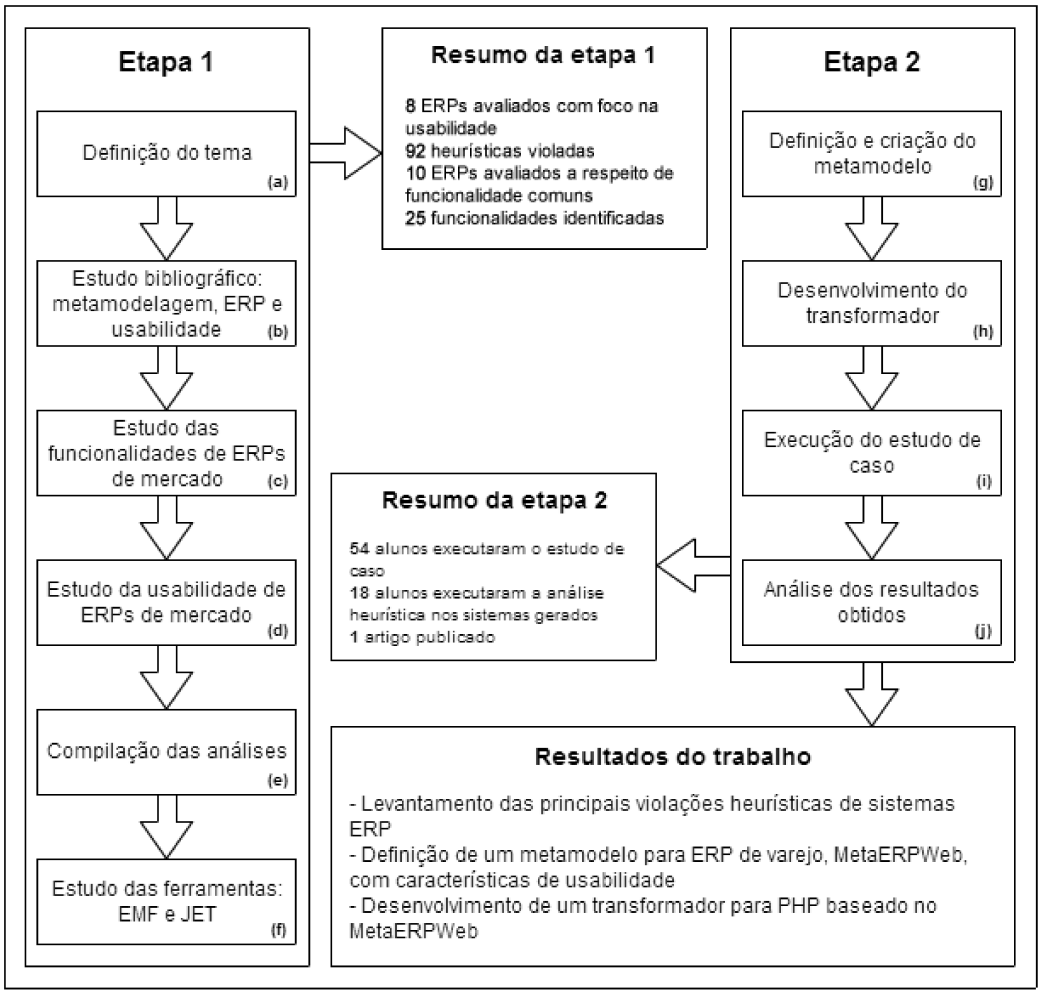

Figura 1.1: Visão geral da metodologia utilizada

Na Etapa 2 houve a definição e criação do metamodelo (g), desenvolvimento do respectivo transformador (h); após a homologação destas implementações foi executado um estudo de caso (i) a fim de validar a proposta deste trabalho. O último passo consistiu na análise dos resultados da fase anterior e a conclusão a validade da proposta (j).

1.4 Contribuições e resultados obtidos

A partir da execução dos passos descritos anteriormente e visando os objetivos propostos pode-se apontar as contribuições e resultados a seguir:

- Levantamento das principais violações de sistemas ERP de varejo;
- Levantamento das principais funcionalidades de sistemas ERP de varejo de mercado;
- Definição de uma nova característica funcional de usabilidade (FUF - *Functional Usability Feature*) identificado durante o levantamento das funcionalidades de ERP de varejo de mercado;

- Definição de um metamodelo para sistemas ERP de varejo com características de usabilidade a partir dos levantamentos realizados anteriormente;
- Desenvolvimento de um transformador para a linguagem PHP do metamodelo definido;
- Validação da proposta deste trabalho através de um estudo de caso e sua posterior análise.

1.5 Organização do trabalho

Esta dissertação está dividida em outros quatro capítulos, além deste capítulo introdutório. O segundo capítulo apresenta os fundamentos e técnicas que são utilizados ao longo deste trabalho. O capítulo seguinte aborda a proposta deste trabalho é explanada apresentando o metamodelo proposto bem como seu transformador. Já o capítulo quatro aborda a validação da proposta realizada através da execução de um estudo de caso seguido por algumas análises. O último capítulo apresenta as conclusões, incluindo as contribuições e limitações bem como possibilidades de trabalhos futuros.

Capítulo 2

Fundamentos e técnicas

2.1 Considerações iniciais

Este capítulo apresenta os principais conceitos, técnicas e trabalhos relacionados nos quais este trabalho se baseia. A próxima seção introduz uma série de fundamentos utilizados e citados ao longo desta dissertação entre eles sistemas ERPs, Web 2.0, desenvolvimento dirigido a modelos e conceitos de interação com o usuário. Já a seção 2.3 os principais trabalhos relacionados à proposta.

2.2 Fundamentos

2.2.1 Sistemas ERP

Um sistema ERP possui funcionalidades que abrangem todas as áreas da organização, realizando todas as ligações necessárias entre elas e registrando essas informações. Essa é uma das causas do alto valor de um ERP e do seu alto tempo de desenvolvimento (MILOSZ et al., 2013).

Esse tipo de software é geralmente dividido em módulos bem definidos que são encarregados de controlar uma área dentro da empresa. Além disso, o termo ERP pode ser muito amplo visto que o sistema deve atender a empresa que o utilizará. Como cada empresa possui uma demanda diferente cada ERP deve atender essas particularidades, além disto os ERPs atendem diferentes domínios como: indústria, distribuição, varejo, entre outros. Entretanto, apesar de suas disparidades, o sistema ERP possui características comuns e as variabilidades são construídas a partir de um núcleo comum.

Um sistema ERP é definido como um sistema de informação, assim utiliza-se a seguinte nomenclatura de Laudon e Laudon (2007):

- Dado: sequência de fatos brutos representando eventos que ocorrem nas organizações ou no ambiente físico antes de terem sido organizados e arranjandos de uma forma que as pessoas possam entendê-los e usá-los;

- Informação: dados apresentados em uma forma significativa e útil para os seres humanos;

- Registro: uma linha da tabela de banco de dados de uma determinada entidade, representa todas as informações acerca daquele item específico da entidade.

2.2.2 Web 2.0

Em 1993 o navegador Mosaic foi apresentado iniciando uma explosão de popularidade para a Web. Esta tecnologia atraiu a atenção de investimentos assim crescendo, ao longo dos anos, de forma intensa; esse período conhecido como ”bolha ponto-com”teve seu término em 2001 quando aconteceu o estouro da bolha e muitas das empresas que estavam neste meio entraram em colapso financeiro.

Até então a Web, conhecida também como Web 1.0 ou Web tradicional, possuia um único fluxo onde o conteúdo era produzido pelo Webmaster e então disponibilizado aos usuários conforme retratado na Figura 2.1.

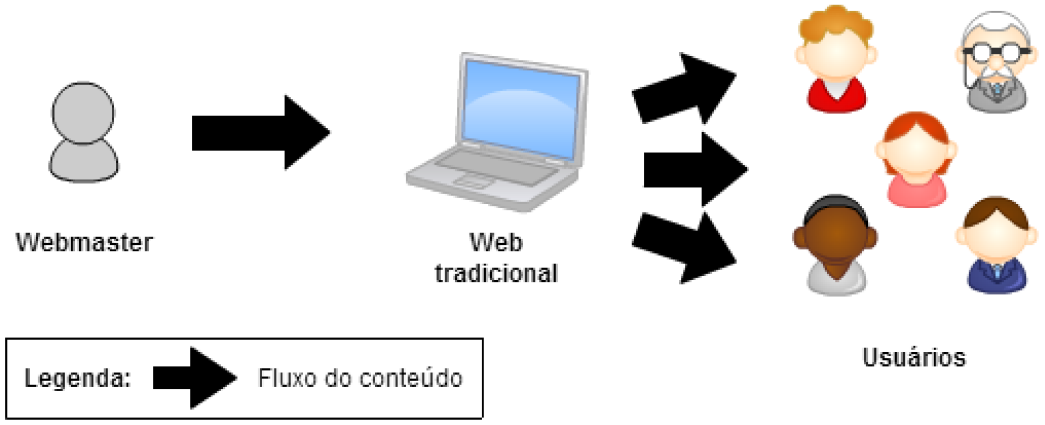

Figura 2.1: Web 1.0 ou Web tradicional

O estouro da bolha marcou um momento decisivo para a Web, enquanto alguns julgavam que a Web tinha sido superestimada a realidade é que esta tecnologia estava próxima de uma revolução. Durante uma sessão de *brainstorm* entre O'Reilly e MediaLive International o conceito de Web 2.0 foi introduzido (O'REILLY, 2005).

Sucintamente pode-se definir a Web 2.0 como uma nova forma de encarar a Web onde os usuários também se tornam responsáveis pelos conteúdos existentes, de acordo com a Figura 2.2, onde o propósito é colaborar e compartilhar conteúdo (DEITEL; DEITEL, 2008).

Contudo suas características vão além da colaboração. É possível definir sete princípios que norteiam as aplicações na Web 2.0 (O'REILLY, 2005):

1. Web como plataforma: a Web torna-se também uma plataforma poderosa sufiente para executar aplicações anteriormente executadas apenas em plataforma *desktop*;

2. Aproveitamento da inteligência coletiva: a aparente razão pela qual gigantes, nascidas na era da Web tradicional, terem sobrevivido a Web 2.0. Sites como Google[1], eBay[2] e

[1]https://www.google.com/
[2]http://www.ebay.com/

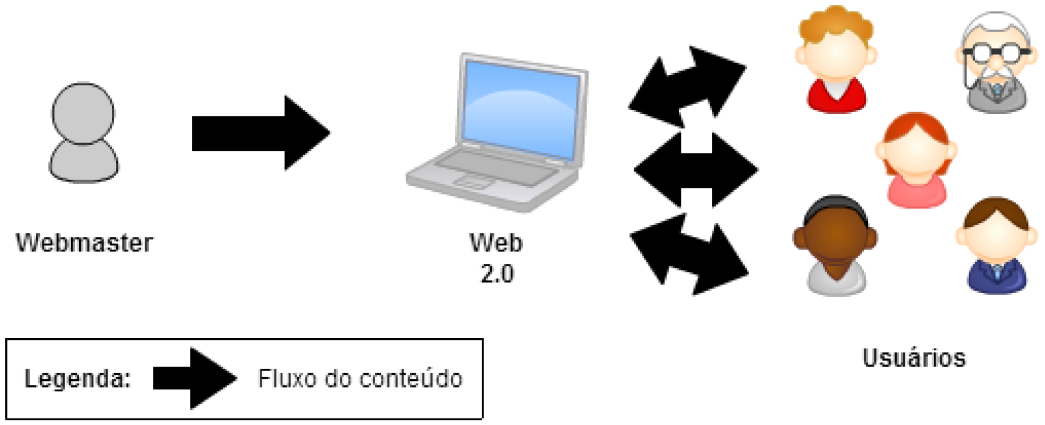

Figura 2.2: Web 2.0

Amazon[3] utilizam a inteligência coletiva para aprimorar seus resultados e oferecer cada vez mais dados e informações interessantes e relevantes aos usuários. Outros sites e aplicações como Wikipedia[4] ou redes sociais utilizam a inteligência dos usuários para criar e aprimorar conteúdos e ligá-los (através de *hyperlinks*) constuindo assim uma rede de informações muito rica;

3. Informação é muito importante: a maioria das aplicações Web relevantes possui informações especializadas. O Google possui informações sobre milhares de páginas Web, a Amazon possui um grande banco de dados de produtos, a base de dados do eBay contém milhares de registros sobre produtos e vendedores;

4. Fim do ciclo de *release* de software: como aplicações Web não precisam ser instaladas pelo usuário e sim apenas acessadas faz-se possível a liberação de serviços ao invés de versões da aplicação. Na Web é possível medir a assiduidade do usuário. Assim este possui um papel importante na melhoria do serviço já que com a análise correta dessa informação os desenvolvedores podem alterar a funcionalidade ou até abandoná-la caso não agrade aos usuários;

5. Modelos de programação mais leves: alguns dos casos bem-sucedidos na Web acontecem pois substituiram a teoria do *hypertext* por técnicas mais simples e práticas. Um dos exemplos é o abandono de implementação de Webservices através SOAP (*Simple Object Access Protocol*) com todo o seu formalismo para uma abordagem mais leve, o REST (*Representational State Transfer*);

6. Software disponível em mais de um dispositivo: como a plataforma é Web a aplicação pode ser acessada através de qualquer dispositivo com acesso a internet independente da plataforma do usuário;

7. Experiência de usuário rica: a tecnologia AJAX possui um papel importante dentro do contexto da Web 2.0 onde permite que experiência do usuário seja bem mais agradável

[3] http://www.amazon.com/
[4] http://www.wikipedia.org/

aproximando-se da mesma experiência que tem em aplicações *desktop*. Esta tecnologia permite que requisições assíncronas possam ser executadas alterando a página visualizada pelo usuário sem a necessidade de recarregá-la totalmente. Dessa forma torna-se mais fácil a migração de sistemas para a Web do ponto de vista de usabilidade e adequação.

Outras definições como Web 3.0, Web 4.0, não invalidam os pontos apresentados para a Web 2.0 uma vez que aqueles complementam este e não redefinem todas as suas características. Dessa forma estas definições são maneiras de encarar a web e destacam a sua evolução (PATTAL; LI; ZENG, 2009).

2.2.3 Desenvolvimento Dirigido a Modelos

A principal característica do Desenvolvimento Dirigido a Modelos (MDD - *Model Driven Development*) é tornar os modelos, ao invés dos códigos, como principal artefato de desenvolvimento, e concentrando nos modelos a maior carga do desenvolvimento e manutenção (CERNY; SONG, 2010). Na abordagem dirigida a modelos os modelos usam os conceitos sem relacionar-se à tecnologia de implementação. Isso torna os modelos mais fáceis de especificar, entender e manter já que são aderentes a um domínio (OBRENOVIC; STARCEVIC, 2005).

Em abordagens tradicionais modelos são apenas usados para documentar a aplicação e, muitas vezes, se tornam obsoletos ao longo do desenvolvimento Normalmente eles não são atualizados ao longo do processo ou, quando são atualizados, muitas vezes se tornam apenas um peso para a equipe de desenvolvimento já que normalmente o modelo é atualizado de acordo com a implementação e não o contrário. Já na abordagem de MDD (FRANCE; RUMPE, 2007) o modelo é um ponto-chave no desenvolvimento e não apenas usado para documentar a aplicação.

A Arquitetura Dirigida a Modelos (MDA - *Model Driven Architecture*), do OMG[5] (*Object Management Group*), descreve os requisitos do sistema pelo Modelo Independente de Plataforma (PIM - *Plataform Indepentent Model*). Nesse modelo apenas as regras de negócio são consideradas, sem detalhes técnicos. Então ocorre a transformação para o Modelo Específico de Plataforma (PSM - *Plataform Specific Model*), onde detalhes técnicos são levados em conta para uma plataforma específica (CHITFOROUSH; YAZDANDOOST; RAMSIN, 2007; ELLEUCH; KHALFALLAH; AHMED, 2007; JESPERSEN; LINVALD, 2003). Assim um PIM pode gerar vários PSM: um para cada plataforma distinta.

Portanto, a partir do modelo uma série de transformações podem ser aplicadas, onde essas transformações são um conjunto de regras que definem como um modelo de entrada deve ser convertido automaticamente (MUKHTAR et al., 2013) em outro modelo (transformação M2M - *Model-to-Model*) ou convertido em código-fonte (transformação M2C - *Model-to-Code*). Os modelos podem ser usados para gerar código parcial ou total, para diferentes tecnologias, assim as tarefas repetitivas são encapsuladas nas transformações (CIRILO et al., 2010) e o desenvolvedor pode se focar na geração do modelo e na complementação do código gerado a partir das transformações.

Os modelos são sintaticamente e semanticamente definidos por metamodelos. Um metamodelo é um instrumento para definição da linguagem do modelo. Assim como um analisador

[5] http://omg.org/

sintático e semântico define as regras que uma linguagem deve seguir o metamodelo define cada elemento que são reconhecidos pelo modelo. Já que o metamodelo é também um modelo ele precisa ser descrito, tendo seu próprio metamodelo que descreve sua semântica. A linguagem de um metamodelo é chamada metalinguagem. A metalinguagem é diferente da linguagem de modelo porque é usada para descrever linguagens de modelo.

A Figura 2.3 representa graficamente, de forma simplificada, o MDD: o metamodelo define as regras que devem ser seguidas pelo desenvolvedor no desenvolvimento do modelo. Então o transformador recebe como entrada o modelo e possui conhecimento das regras do metamodelo para gerar um novo modelo (M2M) ou um código-fonte (M2C).

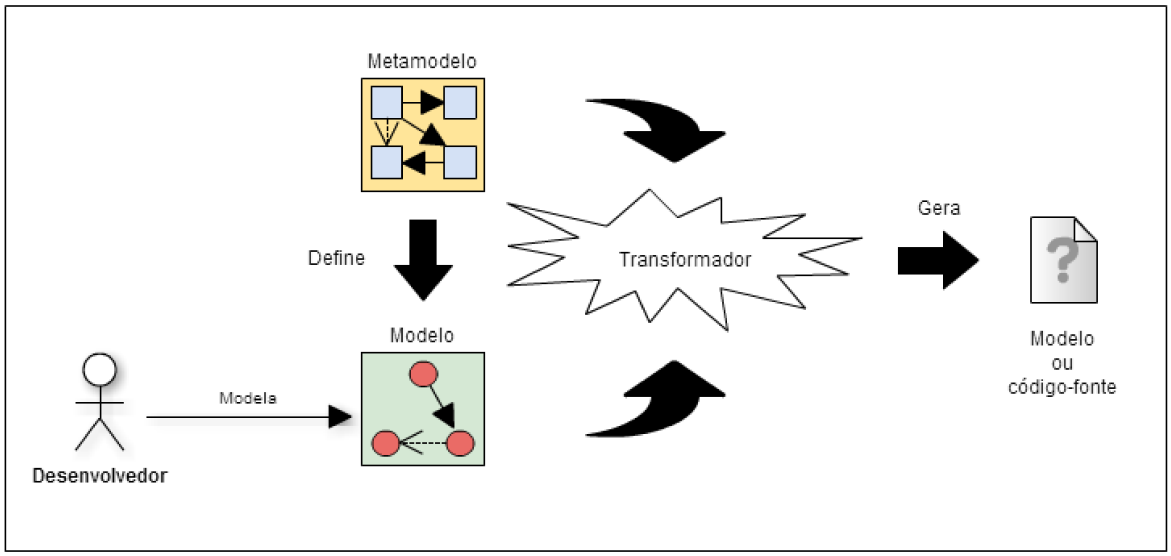

Figura 2.3: Representação gráfica do MDD - adaptado de Lucrédio (2009)

Usualmente o processo de MDD possui duas etapas: engenharia de domínio e engenharia de aplicação. A Engenharia de Domínio (ED) está centrada no desenvolvimento de artefatos[6] para reuso como componentes, transformadores, analises, modelos, etc. A ED está focada em um domínio de sistemas e não em uma solução específica; seu desenvolvimento empenhado em atender características comuns para todos os componentes do domínio em questão e planejar como sistemas individuais poderão ser criados usando seus artefatos (CZARNECKI, 2005).

A Engenharia de Aplicação (EA) é onde as aplicações concretas são construídas usando os artefatos reusáveis criados na etapa de ED. Assim como o planejamento de aplicações tradicionais este é iniciado com o levantamento de requisitos, análise e especificação entretanto os requisitos são especificados na forma de um requisito genérico produzido na ED (CZARNECKI, 2005).

A Figura 2.4 demonstra como o processo de alimentação e retroalimentação funciona onde as duas etapas se alimentam: ED provê a EA os artefatos genéricos reutilizáveis para o uso em sistema específicos, já a EA pode produzir novos requisitos à ED a fim de tornar os artefatos ainda mais genéricos para um número maior de situações em aplicações específicas. Na etapa de ED a análise de domínio refere-se a obter uma série de modelos de domínio a fim de delimitá-

[6]Artefato é um produto gerado a partir de uma atividade

lo expressando a reusabilidade e requisitos configuráveis; a implementação de domínio provê componentes reusáveis, transformadores entre outros (HIDALGA; ZHAO; SAMPAIO, 2008). Já na EA a análise do sistema é a avaliação do sistema de acordo com o planejamento tradicional; a derivação do sistema é o uso dos artefatos reutilizáveis providos pela ED na aplicação específica.

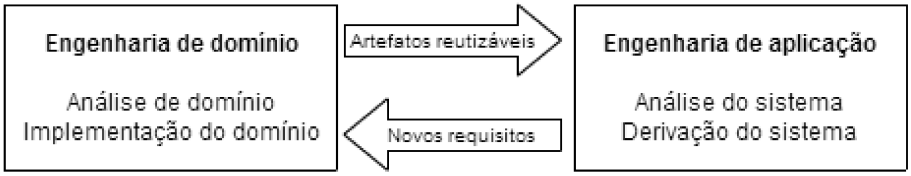

Figura 2.4: Representação gráfica de ED e EA - adaptado de Czarnecki (2005)

2.2.4 Modelagem Específica de Domínio

A UML[7] (*Unified Modeling Language*) é adotado como padrão pelo MDA, já que é um padrão OMG (GAO; LI; ZHENG, 2006; ROBERT et al., 2009). Alguns autores como Kelly e Tolvanen (2008) e Cook et al. (2007), porém, consideram que o UML é muito abrangente e aborda um conjunto diverso de conceitos que se torna difícil de usá-lo. Ao mesmo tempo, a UML é muito rígido para adaptar-se às necessidades específicas de domínio. Enquanto UML lida com todos os tipos de aplicações a Modelagem Específica de Domínio (DSM - *Domain-Specific Modeling*) foca em um grupo pequeno, limitado de domínios (KELLY; TOLVANEN, 2008; ROBERT et al., 2009).

A Modelagem Específica de Domínio caminha junto com o MDD, onde os modelos são criados através de uma Linguagem Específica de Domínio (DSL - *Domain Specific Language*) (SADILEK, 2008). DSLs são linguagens que focam nos conceitos e terminologias do domínio, permitindo que o desenvolvedor descreva a essência do problema com abstrações relacionadas ao domínio específico (DEMIRKOL et al., 2012). Além disso uma DSL é mais fácil de aprender e entender (para o especialista do domínio) do que modelos genéricos (TORSEL, 2013).

A utilização de um modelo específico de domínio permite uma maior abstração na modelagem por parte do desenvolvedor diminuindo os esforços de tradução dos conceitos desse domínio para a solução (CHAVARRIAGA; MACÍAS, 2009).

2.2.5 Interação com o usuário

A área de Interação Humano-Computador (IHC) surgiu no início dos anos 1980 como uma área da Ciência da Computação englobando a Ciência Cognitiva e Engenharia de Fatores Humanos. Antigamente os únicos humanos que lidavam com o computador eram profissionais de TI (Tecnlogia da Informação); após o surgimento dos computadores pessoais (PC) todas as pessoas se tornaram potenciais usuários de computadores. E isto trouxe a tona as deficiências dos computadores em respeito a usabilidade para aqueles que gostariam de utilizá-los como ferramentas de trabalho (CARROLL, 2014).

[7] http://www.uml.org/

Moggridge e Atkinson (2007) em 1984, realizou uma apresentação sobre o assunto e descreveu como "*Soft-face*"a combinação entre software e design de interface de usuário. Eventualmente, após sugestão de Bill Verplank, foi dado o nome de design de interação.

Desde então uma série de definições foram dadas para o que é o design de interação, porém é possível definí-la como uma abordagem para resolver problemas de forma interdisciplinar, holística e direcionada a um profundo entendimento do comportamento, cognição, capacidades, desejos e contexto humano (LOWGREN, 2014; NIELSEN; NORMAN, 2013).

A melhoria na usabilidade de um sistema de software pode ser considerada na MDD, possibilitando a geração de interface que buscam atender melhor interação com o usuário. Motivações incluem auxiliar desenvolvedores sem experiência no desenvolvimento de interfaces, permitindo que eles possam se focar na implementação da funcionalidade e contar com esses sistemas na criação de uma interface com maior qualidade na interação (OBRENOVIC; STARCEVIC, 2005; CIRILO et al., 2011).

Apesar de existirem métodos baseados em modelos muito poderosos para criação de modelos muitos deles não dão suporte para primitivas conceituais de usabilidade. Assim a melhoria na usabilidade acaba sendo, geralmente, incluída manualmente após o código ter sido gerado, diminuindo a eficiência do desenvolvedor, que deve modelar o sistema e, após isso, implementar atributos de usabilidade (PANACH; JURISTO; PASTOR, 2013). Essa prática contradiz a ideia do MDD uma vez que os esforços devem ser concentrados na construção de modelos e a geração de código é delegada aos transformadores.

Jespersen e Linvald (2003) discutem que cada sistema é único exigindo um estilo de modelagem de interfaces diferenciado que, portanto, MDD não tem muito a oferecer. Entretanto existem grupos de sistemas que possuem pontos em comum e permitem que os modelos de interface com o usuário compartilhem um estilo de modelagem em comum. Nesse cenário as promessas do MDD se mantém, podendo ser criado um metamodelo que cobre o domínio de uma aplicação.

Heurísticas de usabilidade

Usabilidade pode ser definida como a forma em que um produto pode ser utilizada por usuários específicos para alcançar objetivos específicos com efetividade, eficiência e satisfação em um contexto específico de uso (STANDARDIZATION, 1998). Outra definição descreve usabilidade como o aprendizado de um sistema, sua eficiência no uso, sua habilidade de evitar e gerenciar erros bem como a satisfação do usuário (NIELSEN, 1993).

Vários benefícios são alcançados a partir da preocupação com a usabilidade como melhoria de produtividade, diminuição de custos com treinamento e documentação entre outros (TRENNER; BAWA, 1998; DONAHUE, 2001; JURISTO; MORENO; SANCHEZ-SEGURA, 2007b).

Um dos métodos para validar e encontrar problemas de usabilidade em interfaces gráficas é conhecido como avaliação heurística (NIELSEN, 1994; NIELSEN; MOLICH, 1990). Especificamente envolve avaliadores examinando a interface e avaliando sua qualidade de acordo com princípios de usabilidade, que são denominados heurísticas (NIELSEN, 1995b).

No ínicio dos anos 90 Jakob Nielsen e Rolf Molich elaboraram dez princípios gerais para desenvolvimento de interfaces, que ficaram conhecidos como Heurísticas de Usabilidade de Nielsen

(NIELSEN, 1995a). Esses princípios pontuam conceitos básicos que garantem uma interface mais amigável e fácil de ser usada, a saber:

H1. Visibilidade do status do sistema: o sistema sempre deve manter o usuário informado sobre o que está acontecendo;

H2. Ligação entre sistema e o mundo real: o sistema deve se comunicar com o usuário de forma que ele entenda, usando palavras, frases e conceitos familiares para o usuário;

H3. Controle do usuário e liberdade: frequentemente usuários fazem escolhas erradas e precisam de uma ”saída de emergência”para sair do estado indesejado;

H4. Consistência e padrões: usuários não devem ter que se preocupar com diferentes palavras, situações ou ações significar a mesma coisa;

H5. Prevenção de erros: mesmo com boas mensagens de erro é prudente evitar que o problema ocorra. Pode-se eliminar condições passíveis de erro ou verificar e apresentar ao usuário com uma confirmação antes de seguir com a ação;

H6. Reconhecimento ao invés de lembrança: o usuário não deve ter a memória acionada todo o tempo, o sistema deve manter objetos, ações e opções visíveis;

H7. Flexibilidade e eficiência no uso: o sistema deve ser fácil para usuários leigos, mas também adaptar-se a usuários experientes, permitindo que ações repetitivas sejam acessadas facilmente, por exemplo;

H8. Estética e design minimalista: diálogos não devem conter dados irrelevantes;

H9. Recuperação de erros: mensagens de erro devem ser expressas em linguagem plana (não apresentar códigos) que indiquem o problema e sugira uma solução;

H10. Ajuda e documentação: apesar de ser melhor que um sistema possa ser usado sem documentação, pode ser necessário fornecer ajuda e documentação.

Apesar de questões de usabilidade impactarem principalmente na interface com o usuário muitos pontos impactam no núcleo funcional do sistema (SEFFAH; METZKER, 2004; BIAS; MAYHEW, 2005). Por exemplo, para a H1 - *status* do sistema além da interface ter um espaço para exibir a notificação, é necessário existir a funcionalidade que analise a condição do sistema e que traduza esse estado para ser exibido corretamente ao usuário.

Dessa forma verifica-se que as questões de usabilidade não devem ser abordadas apenas ao final do desenvolvimento e sim nas suas fases iniciais prevenindo que problemas funcionais impactem negativamente na usabilidade (JURISTO; MORENO; SANCHEZ-SEGURA, 2007b).

2.3 Trabalhos relacionados

Cirilo et al. (2010) propuseram um processo dirigido a modelos para construção de interfaces ricas para aplicações ubíquas[8]. No trabalho os autores discutem uma abordagem baseada em modelos e transformações para criação de interfaces para diversos dispositivos. O trabalho também inclui um adaptador de interfaces em tempo de execução tratando a interface para cada dispositivo específico que acesse a aplicação. A ideia principal da proposta é contar com apenas um modelo que gere interfaces para diversos dispositivos onde a decisão de qual interface ser utilizada deve ser tomada em tempo de execução da interface adaptando-a conforme o dispositivo que a acessa. Após a conceituação do processo, implementação do metamodelo e seu tranformador os autores iniciaram a validação do processo. Com a análise dos resultados é possível observar uma redução no tempo de prototipagem das versões estáticas da interface advinda do emprego das transformações M2C para geração do código a partir do modelo criado pelos participantes do estudo de caso proposto durante a validação.

Trias (2012) identificou que os modelos já existentes não atendem adequadamente a geração de sistemas CMSs (*Content Management Systems*) e propõe um novo metamodelo especificamente para esse tipo de sistema. Foram levantadas nove características que diferenciam um CMS de uma outra aplicação Web, algumas são: a responsabilidade de atualizar o conteúdo passa a ser dos seus autores e não mais do *Webmaster*, o conteúdo das páginas é categorizado, os conteúdos são gerenciados e não as páginas. Além disso há uma independência entre conteúdo e apresentação. A metodologia proposta foi divida em duas fases: resolução e validação, cada uma delas sub-dividas em outras tarefas. Na primeira tarefa da fase de resolução foi feita uma análise de plataformas CMS disponíveis no mercado onde foi concluído quais as plataformas mais populares. Em seguida foi realizada uma definição dos metamodelos de cada plataforma onde foram identificados alguns elementos chave no domínio. Assim um metamodelo comum foi definido e implementado tendo como base os três metamodelos da tarefa anterior. O metamodelo criado foi divido em quatro pontos: navegação, conteúdo, usuário e comportamento. A navegação considera elementos que definem a estrutura de navegação e aqueles que permitem a navegação entre essas estruturas. O segundo ponto captura as informações gerenciadas pelo CMS; o seguinte define a permissão de cada usuário. Já o comportamento contém elementos que definem diferentes funções que podem ser feitas através de sistemas CMS. Os autores apresentaram a validação da proposta que ocorreu em paralelo com a fase de implementação onde foi feita a validação dos metamodelos das plataformas, validação do metamodelo comum e validação da implementação deste último. Basicamente em cada tarefa foi gerado um modelo em cima dos metamodelos propostos.

Cerny e Song (2010) propõem uma solução para criação de modelos que gerem automaticamente interfaces de formulários com validação de dados a partir da definição de regras. No trabalho é clara a preocupação com a geração de uma interface completa, que não necessite da intervenção do desenvolvedor pós-geração. Essa preocupação é justificada já que a manutenção de uma interface auto-gerada é alta devido a sua complexidade podendo não justificar sua adoção. A solução dos autores é uma extensão da UML que permite que modelos sejam capazes

[8]Computação Ubíqua é entendida como a computação onipresente, ou seja, em vários tipos de disposivos e todo o tempo (PENDYALA; SHIM, 2009)

de armazenar dados adicionais para validação completa e geração de formulários para interfaces ricas. A proposta também elimina o trabalho manual do desenvolvedor na manutenção da interface, que precisa apenas se preocupar apenas com o modelo em questão.

Juristo, Moreno e Sanchez-Segura (2007b) estudaram diversos trabalhos e identificaram características funcionais que tiveram alto impacto na usabilidade do software. A partir desse estudo os autores elaboraram pontos denominados FUFs (*Functional Usability Feature*), que foram derivados de heurísticas, regras e princípios de usabilidade. Os FUFs são definidos como recomendações para melhorar a usabilidade do sistema que possui um impacto no planejamento arquitetural da aplicação (PANACH; JURISTO; PASTOR, 2013). Os FUFs foram definidos como:

1. *Feedback*: informar os usuários o que está acontencendo com o sistema;
2. Desfazer: desfazer ações do sistema em vários níveis;
3. Cancelar: cancelar a execução de um comando ou de uma aplicação;
4. Validação de formulário e campo: melhorar a entrada de dados para os usuários e correção via software o mais rápido possível;
5. Wizard: ajuda para executar tarefas que exigem diferentes passos com entrada de dados pelo usuário;
6. Perícia do usuário: adaptar as funcionalidades do sistema de acordo com a perícia do usuário;
7. Ajuda multi-nível: fornecer ajuda em diferentes níveis para usuários diferentes;
8. Uso de linguagens diferentes: o usuário deve poder trabalhar usando a linguagem própria, moeda corrente, formato de código postal, formato de data, etc...
9. Alerta: avisar o usuário sobre uma ação com consequências importantes.

Os FUFs foram gerados a partir de padrões de interação encontrados na literatura porém com o objetivo de ser mais simples além de demonstrar como características de usabilidade afetam a arquitetura do sistema. Os autores tinham como objetivo que os FUFs fossem incorporados no processo de desenvolvimento como requisitos funcionais (PANACH et al., 2015).

Em outro trabalho Panach et al. (2015) complementam o conceito de FUF tornando-o aplicável em cenários de desenvolvimento dirigido a modelos. Cada FUF pode ser dividido em mecanismos de usabilidade (UM - *Usability Mechanism*), cada UM pode ser especificado em MoUs (*Mode of Use*) e, cada MoU, alcançado através das propriedades. A Figura 2.5 apresenta graficamente essa especificação.

A seguir apresenta-se como cada FUF é dividido em mecanismos de usabilidade e seus respectivos objetivos (JURISTO; SANCHEZ-SEGURA, 2006; JURISTO; MORENO; SANCHEZ-SEGURA, 2007a). É importante observar que, em relação a proposta original dos FUFs houve diluições de pontos (por exemplo, o item Alerta passou a fazer parte do item *Feedback*) além da criação de novos itens:

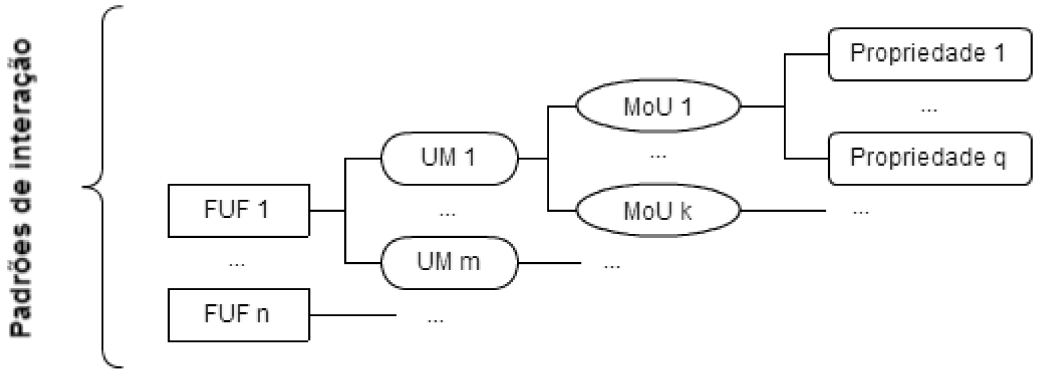

Figura 2.5: Especificação de FUFs - adaptado de Panach, Aquino e Pastor (2014)

FUF1. *Feedback*

(a) Status do sistema: informar o usuário sobre o status interno do sistema

(b) Interação: informar o usuário que sua ação foi registrada pelo sistema

(c) Alerta: informar o usuário sobre qualquer ação com importantes consequencias

(d) *Feedback* de ações demoradas: informar o usuário que sua ação está sendo processada pelo sistema de levará algum tempo para completar

FUF2. Desfazer / Cancelar

(a) Desfazer global: desfazer ações no sistema em vários níveis

(b) Desfazer de objeto específico: desfazer ações no sistema em um objeto específico

(c) Abortar operações: cancelar a execução de uma ação ou da aplicação em geral

(d) Voltar: voltar para um estado em particular em uma sequencia de execução de comandos

FUF3. Prevenção e correção de erros de preenchimento

(a) Entrada de texto estruturado: prevenir que o usuário preencha dados de forma errada

FUF4. *Wizard*

(a) Execução passo-a-passo: ajudar usuário a realizar tarefas que exigem diferentes passos com correção de preenchimento de dados

FUF5. Perfil de usuário

(a) Preferências: salvar a opção de cada usuário ao usar funções do sistema

(b) Espaço de objetos pessoais: salvar a opção de cada usuário ao usar a interface do sistema

(c) Favoritos: salvar certos lugares de interesse do usuário

FUF6. Ajuda

(a) Ajuda multi-nível: prover diferentes tipos de ajuda para diferentes usuários

FUF7. Agregação de comandos

(a) Agregação de comandos: apresentar a possibilidade de realizar certas ações dentro do sistema através de comandos que podem ser construídos a partir de partes menores

Assim observa-se que um FUF pode ser dividido entre vários UMs que são diferentes subtipos do FUF. Ou seja, cada FUF possui um objetivo principal que pode ser espcializado em objetivos mais detalhados, que são os MoUs (PANACH; JURISTO; PASTOR, 2013). Um MoU é um mecanismo utilizado para se atingir um UM pois cada mecanismo de usabilidade pode atingir seu objetivo de forma diferente. Cada MoU atinge um alvo específico que faz parte do objetivo geral do FUF em questão. Diferentes MoUs que fazem parte do mesmo FUF têm o mesmo objetivo sem conflitar um com o outro.

Por exemplo, o UM 1.1, "status do sistema", do FUF1 *feedback* tem como objetivo informar o usuário sobre o *status* interno do sistema. É possível identificar, ao menos, três modos (MoUs) de alcançar esse objetivo: informar sobre o sucesso ou falha da execução (MoU1.1.1); mostrar a informação salva no sistema (MoU1.1.2); e mostrar o status de ações relevantes (MoU1.1.3). O primeiro MoU fornece a informação se o sistema aceitou a ação do usuário ou não; o MoU1.1.2 visa exibir o *status* do sistema usando informação salva no repositório. Por fim, o último MoU é concebido para apresentar o status indicando quais ações podem ser disparadas em quais momentos. Maiores detalhes sobre os UMs e seus MoUs podem ser encontrados em Juristo, Moreno e Sanchez-Segura (2007a).

Cada MoU possui várias configurações que satisfazem o seu objetivo; estas são chamadas de propriedades. Usando o MoU1.1.1 do exemplo anterior identifica-se duas propriedades: seleção de serviço (Prop1.1.1.1) e visualização de mensagem (Prop1.1.1.2). A primeira propriedade existe pois cada ação deve informar se foi executada com sucesso ou não; já a última é definida pois é necessário estabelecer onde a mensagem será exibida para o usuário.

Existem casos que os analistas precisam adaptar as propriedades para o sistema em desenvolvimento e, em outros casos, isso não é necessário. Assim definem-se as propriedades configuráveis e as propriedades não configuráveis, respectivamente.

As propriedades configuráveis exigem que o analista tome decisões sobre como elas devem ser elaboradas. Por exemplo, na propriedade Prop1.1.1.2 é necessário definir onde a mensagem será exibida e essa decisão deve ser tomada pelo analista. Já as propriedades não configuráveis possuem condições inalteradas independente do sistema. A propriedade Prop1.1.1.1, por exemplo, é uma propriedade não configurável já que o sistema deve informar o sucesso ou falha em cada execução de uma ação.

2.3.1 Análise comparativa

O trabalho apresentado nesta dissertação baseia-se em diversas características dos trabalhos descritos anteriormente. Entretanto são apresentadas contribuições próprias que complementam os trabalhos correlatos. De forma geral os trabalhos apresentam soluções orientadas a modelos para objetivos específicos e também questões de usabilidade.

O *Model Driven RichUbi* (CIRILO et al., 2010) (1) apresenta uma solução orientada a modelos para o desenvolvimento de interfaces para diversos dispositivos, sem preocupação com características funcionais da aplicação nem de usabilidade. Já Trias (2012) (2) criaram um metamodelo para desenvolvimento de sistemas CMS, porém, também, sem foco em características de usabilidade. Cerny e Song (2010) (3) apresentam um metamodelo para criação de interfaces especificamente para formulários de preenchimento. Os FUFs (PANACH et al., 2015) (4) são expostos como regras para a criação de interfaces com foco em usabilidade e preocupando-se com as funcionalidades necessárias para que as características funcionais sejam possíveis; completando Panach et al. (2015) (5) apresentam como estas características podem ser aplicadas em soluções orientadas a modelo.

Cada um destes trabalhos podem ser comparados a proposta desta dissertação através de algumas características, que estão apresentadas na Tabela 2.1; nela as colunas identificadas com números estão associados aos identificadores do parágrafo anterior.

Tabela 2.1: Comparativo entre os trabalhos relacionados e o MetaUsaERPWeb

Característica	1	2	3	4	5	MetaUsaERPWeb
Solução orientada a modelos	X	X	X		X	X
Preocupação com usabilidade				X	X	X
Preocupação com funcionalidade	X	X				X
Domínio não especificado	X			X	X	
Domínio de sistemas CMS		X				
Domínio de sistemas ERP						X

Este trabalho une os conceitos de desenvolvimento dirigido a modelos com geração (através de um transformador automatizado) de código funcional com características de usabilidade. Esta combinação não é abordada pelos trabalhos relacionados presentes nesta seção.

2.4 Considerações finais

Este capítulo apresentou uma revisão da literatura a respeito dos principais conceitos e técnicas envolvidos no trabalho elaborado além dos trabalhos relacionados. Conceitos de sistemas ERPs bem como de modelagem de aplicações e princípios de usabilidade foram abordados. Todos esses conceitos e técnica serão aplicados neste trabalho.

Capítulo 3

MetaUsaERPWeb: abordagem proposta

3.1 Considerações iniciais

A produtividade do usuário depende da sua própria eficiência, com isso garantir a qualidade da interface gráfica com o usuário é um dos elementos chave para melhorar a eficiência do usuário e, como consequência, a eficiência e consistência dos dados do ERP (HOLSAPPLE; WANG; WU, 2006; MILOSZ et al., 2013).

Um ERP pode ser caracterizado como um domínio específico, ou seja, agrupa-se em um conjunto de sistemas que apresentam funcionalidades semelhantes. Um ERP que atende um determinado segmento pode ser considerado uma Linha de Produto de Software (SPL - *Software Product Line*), onde um conjunto de artefatos reutilizáveis é construído a partir de características comuns dos sistemas e que se diferenciam um do outro por um ou mais pontos, denominados de variabilidade (CLEMENTS; NORTHROP, 2001). Nesse viés, pode-se observar que muitas das funcionalidades semelhantes estão relacionadas a intensa entrada de dados por parte do usuário.

Baseando-se nos pontos descritos este trabalho propõe o desenvolvimento do MetaUsaERPWeb para dar suporte ao desenvolvimento de um sistema ERP na Web no domínio de varejo considerando as características funcionais de usabilidade que são aderentes a estes sistemas.

A Seção 3.2 aborda o estudo do domínio onde são apresentados diversos estudos e comparações que formam a base para a definição da proposta; a Seção seguinte, 3.3, apresenta o MetaUsaERPWeb e seus detalhes. Por fim a Seção 3.4 apresenta algumas considerações finais referentes a este capítulo.

3.2 Estudo do domínio

Além do estudo da literatura apresentada na seção anterior este trabalho também realizou uma série de buscas acerca do assunto a fim de possuir alicerces mais sólidos para propor uma solução mais assertiva e significativa para os domínios, principalmente, de MDD e ERP.

A próxima subseção expõe a pesquisa exploratória realizada em cima de sistemas ERP do mercado a fim de identificar suas semelhanças e principais falhas. Na sequencia é realizada uma

releitura do conceitos de FUFs associando-os às Heurísticas de Nielsen que servirão de base para todas as análises seguintes nesta dissertação.

3.2.1 Pesquisa exploratória

Com o objetivo de identificar semelhanças e falhas comuns em sistemas ERPs de varejo foi realizada uma pesquisa exploratória. A pesquisa foi feita em dois tipos de análise distintas: uma focando nos aspectos de usabilidade e outra nos aspectos funcionais.

Análise heurística de usabilidade

A primeira análise foi realizada pelos alunos de graduação e mestrado do curso de Ciência da Computação da Universidade Federal de São Carlos (UFSCar) - *campus* Sorocaba ao longo do segundo semestre de 2014 na disciplina de Interface Humano-Computador. O objetivo dessa pesquisa era realizar uma análise heurística e listar as principais violações heurísticas cometidas pelos sistemas ERPs de varejo. Os avaliadores examinam a interface e determinam sua qualidade utilizando alguns princípios de usabilidade, neste caso, as heurísticas de Nielsen. Quinze avaliadores, divididos em quatro grupos, realizaram a análise de uma área do sistema; cada grupo avaliou, em média, dois sistemas distintos. Oito sistemas foram selecionados para a análise: GestãoJá[1], wEstoque[2], WK ERP Lite[3], ERP Next[4], WebERP, ERP Now[5], TinyERP[6] e Bling[7]. A seleção utilizou dois critérios: disponibilidade de acesso gratuito e sistemas exclusivamente Web.

A Tabela 3.1 apresenta a quantidade consolidada de violações por heurística nessa análise.

Tabela 3.1: Heurísticas consolidadas violadas por oito ERPs de mercado

Heurística	Número de violações	Porcentagem
H1	16	17,39%
H2	6	6,52%
H3	9	9,78%
H4	11	11,96%
H5	19	20,65%
H6	5	5,43%
H7	2	2,17%
H8	8	8,70%
H9	8	8,70%
H10	8	8,70%

Observa-se, portanto, que a heurística mais violada é a H5 (prevenção de erros) com 20,65%

[1]https://gestaoja.com.br/
[2]http://www.westoque.com.br/
[3]http://www.erplitefree.com.br/
[4]https://erpnext.com/
[5]https://www.erpnow.com.br/
[6]https://www.tiny.com.br/
[7]https://www.bling.com.br/

do total de violações, seguida pela H1 (visibilidade do status do sistema) e por H4 (consistência e padrões) com 17,39% e 11,96% respectivamente.

Já a Tabela 3.2 exibe as violações por sistemas. O número relativo de violações do sistema é mostrado entre parênteses ao lado do número absoluto. O nome dos sistemas foi abreviado e sua referência é exposta na Tabela 3.3.

Tabela 3.2: Heurísticas detalhadas violadas por oito ERPs de mercado

Heurística	GJ	wE	WK	ENe	WE	ENo	TN	BL
H1	1 (7%)	3 (18%)	2 (18%)	5 (45%)	1 (7%)	1 (13%)	2 (33%)	1 (10%)
H2	2 (13%)	1 (6%)	1 (9%)	0 (0%)	1 (7%)	0 (0%)	0 (0%)	1 (10%)
H3	1 (7%)	1 (6%)	1 (9%)	2 (18%)	2 (14%)	0 (0%)	1 (17%)	1 (10%)
H4	3 (20%)	3 (18%)	0 (0%)	0 (0%)	0 (0%)	2 (25%)	1 (17%)	2 (20%)
H5	3 (20%)	6 (35%)	1 (9%)	3 (27%)	3 (21%)	2 (25%)	0 (0%)	1 (10%)
H6	0 (0%)	2 (12%)	2 (18%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (10%)
H7	1 (7%)	0 (0%)	1 (9%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
H8	2 (13%)	0 (0%)	1 (9%)	0 (0%)	3 (21%)	0 (0%)	1 (17%)	1 (10%)
H9	1 (7%)	1 (1%)	1 (9%)	0 (0%)	2 (14%)	1 (13%)	1 (17%)	1 (10%)
H10	1 (7%)	0 (0%)	1 (9%)	1 (9%)	2 (14%)	2 (25%)	0 (0%)	1 (10%)
Total	15	17	11	11	14	8	6	10

Tabela 3.3: Legenda para a Tabela 3.2

Sigla	Nome
GJ	GestãoJá
wE	wEstoque
WK	WK ERP Lite
ENe	ERP Next
WE	WebERP
ENo	ERP Now
TN	TinyERP
BL	Bling

Desta forma podemos assumir que as heurísticas mais violadas são H1, H4 e H5. A H1 diz respeito ao *feedback* onde o usuário deve ser alertado sobre a situação atual do sistema. Dentro desta heurística as principais violações foram identificadas quando o sistema não retornava nenhuma informação a respeito do resultado da ação anterior. Por exemplo, ao enviar um formulário preenchido com uma série de dados a tela seguinte não informava se estes foram salvos ou se aconteceu algum erro. Outro exemplo é a falta de indicações de campos obrigatórios do formulário fazendo com que o usuário não saiba o motivo do mesmo não ser enviado. Carregamentos sob demanda (onde apenas uma parte do conteúdo - tela - é atualizado) também violaram esta heurística; por exemplo, buscas em alguns sistemas retornava apenas uma tela (ou parte da tela) vazia não indicando se aconteceu algum erro na busca ou se não existem registros que a satisfaça.

Consistência e padrões é o foco da H4 cujo objetivo é garantir que palavras e ações diferentes não signifiquem a mesma coisa. As ocorrências mais comuns são o uso de termos diferentes usados para a mesma situação. Por exemplo, a tela de cadastrar e editar possuem os mesmos campos porém o rótulo desses campos é diferente (e o dado é o mesmo). Outro exemplo é o uso indiscriminado de cores sendo utilizada uma para cada situação e alterando-as: uso da cor verde para indicar sucesso em uma ação, em outro momento é utilizada a cor azul e até alterando seu significado utilizando essas cores para indicar erros ou falhas. Também acontece do padrão ”desabilitado”(com o elemento desfocado por exemplo) ser utilizado em apenas alguns itens e não em outros dando a falsa impressão de que existe alguma ação possível naquele elemento.

Por fim, a H5 deve promover a prevenção de erros. É completamente plausível que ERPs possuam mais violações nesta heurística já que esse tipo de sistema lida com um volume grande de informações complexas. É comum que o usuário cometa erros no momento de entrar com essas informações bem como manipular esses dados. Se o sistema realizar as validações necessárias poderá tornar a experiência do usuário mais rica. Tanto indicando o caminho certo a ser seguido, e evitando que perca tempo em atividades repetitivas, como preencher novamente um formulário por conta de algum erro não notado a tempo.

As principais violações da H5 são falta de indicações de como preencher alguns dados de formulário, falta de indicação de itens de preenchimento obrigatório bem como a ausência de verificações de consistência de informações; por exemplo, em alguns casos a data de entrega de um produto poderia ser anterior a data da venda. Em casos extremos a falta de avisos para prevenção dos erros, em formulários de entrada de dados, causava a perda completa das mesmas obrigando o usuário a informá-las novamente de forma integral.

Durante este estudo observou-se que o padrão de autocompletar, onde conforme o usuário preenche determinado campo alguns outros já são preenchidos automaticamente, o que pode prevenir erros (WELIE, 2008).

A Figura 3.1 representa esta funcionalidade. Neste exemplo o usuário está preenchendo um pedido de venda (Figura 3.1(a)), ao informar o produto vendido o sistema automaticamente preenche o campo de preço com o valor já cadastrado no banco de dados (Figura 3.1(b)). O usuário tem a possibilidade de alterá-lo (já que pode existir preço promocional). Isto facilita a interação do usuário pois não precisa lembrar-se (ou ir procurar) o valor padrão de venda do produto.

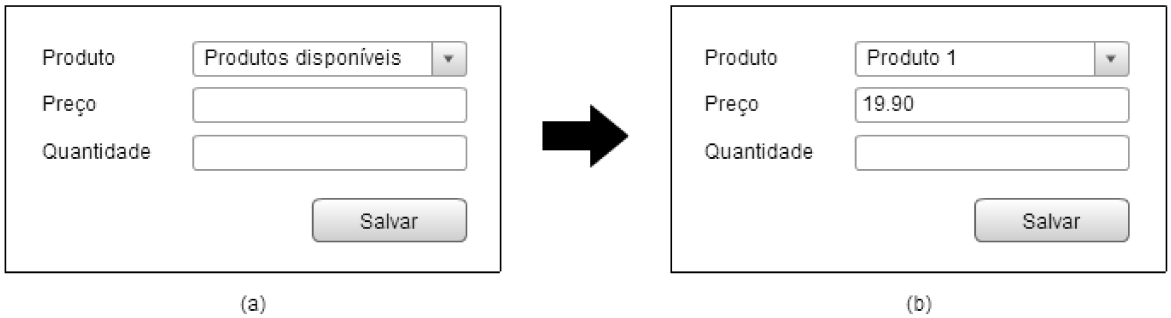

Figura 3.1: Wireframe do padrão de autocompletar

Esta funcionalidade também contribuiria para a heurística H6, reconhecimento ao invés de

lembrança, já que o usuário não é obrigado a lembrar de informações que existem no sistema. Entretanto nenhum mecanismo de usabilidade (MoU) foi encontrado explicitamente. É possível incluí-lo no FUF6 já que é uma ajuda do sistema para o usuário preencher corretamente e com mais velocidade o formulário.

Análise funcional

A segunda análise teve o objetivo de levantar as funcionalidades comuns que os ERPs de varejo de mercado ofereciam a fim de corroborar os alicerces deste projeto.

Para esta pesquisa os seguintes sistemas ERPs foram estudados: GestãoJá, SalesForce[8], ContaAzul[9], eGestor[10], Odoo[11], Intuit QuickBooks Online[12], wEstoque, ERPNext, TinyERP e Bling. Esses sistemas foram escolhidos pois todos disponibilizaram período de *trial* para degustação do sistema além serem baseados na plataforma Web e terem viés de varejo.

Observou-se algumas semelhanças entre todos os sistemas: eles podem ser divididos e agrupados em funcionalidades, denominadas módulos, esses módulos lidam com as suas informações integrando-se uns aos outros. Por exemplo, o módulo de estoque lida com informações dos produtos e sua quantidade disponível, porém o módulo de venda precisa dessa informação para garantir a consistência do estoque (não permitindo estoque negativo) e também precisa comunicar ao módulo financeiro para criar os lançamentos. Outro exemplo é o módulo de clientes que se integra ao módulo de oportunidade para a criação de pré-vendas para o cliente em questão. Na sequência apresenta-se sucintamente cada sistema.

O GestãoJá apresenta em sua versão de *trial* mais completa as opções de controle sobre a comercialização de produtos, a prestação de serviços (venda e acompanhamento de ordens de serviço) e o controle financeiro totalizando quatorze módulos. Apesar de cada um lidar com uma área específica (estoque, prestação de serviço, venda, compra, financeiro, faturamento, etc) todos são integrados trocando informações entre si a fim de otimizar a comunicação entre as diferentes áreas da empresa. Além dos controles já citados o GestãoJá permite a emissão de nota fiscal, geração de boleto.

Após análise do SalesForce concluiu-se que o mesmo trata de um poderoso gerenciador de relacionamento com o cliente, ou seja um CRM (*Customer Relationship Management*). Assim como o exemplo anterior também possui integração com módulo de negociação a fim de garantir regras comerciais definidas pelo CRM sejam respeitadas no momento da negociação e venda final.

Na sua forma mais completa, possuindo controle sobre venda de produto e serviço, o ContaAzul totaliza 14 módulos. Todos os módulos são responsáveis por áreas específicas dentro da empresa porém se integram garantindo o funcionamento do sistema (e da empresa) como um todo. Além do controle sobre venda de produto e serviço o sistema também permite a emissão de nota fiscal, geração de boleto e controle financeiro.

[8]http://www.salesforce.com/
[9]https://contaazul.com/
[10]https://www.egestor.com.br/
[11]https://www.odoo.com/
[12]https://global.intuit.com/quickbooks-online/sui/pt/sui-widget-start-pt.jsp

O software eGestor conta com 13 módulos permitindo o controle de venda de produtos e de serviços, produção de produtos, controle financeiro com emissão de boleto além de emissão de nota fiscal. Assim como os exemplos anteriores os módulos se integram desde a disponibilização de informações até a geração de gatilhos em outros módulos.

Por sua vez o Odoo funciona através de módulos "instaláveis", ou seja, cada módulo pode ser incluído ou não ao ERP criando-se assim um sistema mais próximo possível da realidade da empresa. Tratando-se de um software estrangeiro não existe a possibilidade de emissão de nota fiscal ou geração de boleto. Dependendo dos módulos instalados também acontece a integração de informação entre módulos.

O Intuit QuickBooks Online possui mais de 700 mil empresas o utilizando em todo o mundo, por não se tratar de um software brasileiro não possui as funcionalidades de emissão de nota fiscal e boleto. O sistema conta com 7 módulos mantendo-se no controle de vendas, financeiro e de funcionários. Como de praxe os módulos se comunicam a fim de facilitar a integração entre as diferentes áreas da empresa.

Nos testes identificou-se oito módulos disponíveis no software wEstoque que se foca no gerenciamento de vendas inclusive com um modo de PDV habilitado que permite controle de caixa com sua abertura e sangria (fechamento e conferencia de valores). O sistema possui controle de funcionários e a possibilidade de controle de matriz e filiais. Existe também a integração entre os módulos.

O estrangeiro ERP Next possui, em sua versão de testes, nove módulos. Além de controle de estoque, venda de produto e financeiro o software também possui módulo para Recusos Humanos, gestão de projetos e PCP (Planejamento e Controle de Produção). Assim como os concorrentes o sistema é integrado em diferentes níveis através dos módulos existentes.

A versão gratuita do software TinyERP possui apenas quatro módulos ativos, contando apenas com o controle de clientes, fornecedores, finanças e agenda. A integração entre os módulos existe apenas com o compartilhamento de informações, isso acontece visto que os módulos existentes nessa versão não são muito poderosos para outras possibilidades de integração.

Por fim o Bling possui seis módulos disponíveis na sua versão gratuita, existindo o controle de estoque, venda de produtos e serviços (bem como acompanhamento de ordens de serviço) além do controle financeiro. Essa versão não permitiu emissão de nota fiscal ou geração de boletos, porém possui integração entre os módulos garantindo seu funcionamento.

A Tabela 3.4 apresenta todos os módulos identificados durante a análise dos softwares; as células com X indicam que o módulo em questão está presente no sistema, caso a célula esteja vazia é a indicação de que o módulo não está no escopo. O nome dos sistemas está sendo exibido em siglas, seu significado está presente na Tabela 3.5.

Apesar desta pesquisa ter sido direcionada para ERPs de varejo alguns deles apresentam funcionalidades que vão além deste escopo; esses módulos estão indicados com um asterisco (*) ao lado da sua identificação. Uma breve explicação é feita a seguir:

- Agenda*: Controle de compromissos e *follow-up*;
- Cliente: Cadastro de cliente e seu gerenciamento (CRM);

Tabela 3.4: Relação de módulos disponíveis para cada ERP

Módulo	GJ	SF	CA	eG	oD	QB	wE	ENe	TN	BL
Agenda		X			X			X	X	
Cliente	X	X	X	X	X	X	X	X	X	X
Compra	X		X	X	X	X		X		X
Contrato		X	X		X					
Cotação	X		X		X					
Fabricante	X									
Financeiro	X		X	X	X	X	X		X	X
Fornecedor	X		X	X		X	X		X	
Funcionário					X	X	X	X		
Lead		X			X					
Mailing					X					
NFe	X		X	X						
NFSe	X		X	X						
Oportunidade		X								
Orçamento	X		X			X	X			X
Ordem de produção				X	X			X		
Ordem de serviço	X		X	X						
Produto	X	X	X	X	X		X	X		X
Projeto		X			X			X		
Serviço	X		X	X						
Tarefa				X				X		
Transportadora	X		X	X	X					
Troca							X			
Venda	X		X	X	X	X	X	X		X
WMS					X					
Total	14	7	14	13	15	7	8	9	4	6

Tabela 3.5: Legenda para a Tabela 3.4

Sigla	Nome
GJ	GestãoJá
SF	SalesForce
CA	ContaAzul
eG	eGestor
Od	Odoo
QB	Intuit QuickBooks Online
wE	wEstoque
ENe	ERP Next
WE	WebERP
TN	TinyERP
BL	Bling

- Compra: Gerenciamento de compras de produtos, geralmente integrando com estoque e financeiro;
- Contrato*: Módulo integrante do CRM onde controla o contrato dos clientes com a empresa;
- Cotação: Na necessidade de compra de novos produtos é possível realizar uma cotação com os fornecedores escolhidos através deste módulo;
- Fabricante: Gerenciamento dos fabricantes dos produtos;
- Financeiro: Controle de contas a receber, contas a pagar, categorias financeiras, contas contábeis, contas bancárias, centros de custo, etc;
- Fornecedor: Gerenciamento dos fornecedores dos produtos;
- Funcionário*: Desde controle dos dados dos funcionários até cálculo salário, comissão, folha;
- Lead*: Cliente em potencial que está sendo sondado pela equipe comercial;
- Mailing*: Envio de mail marketing;
- NFe: Emissão de nota fiscal eletrônica de produto;
- NFSe: Emissão de nota fiscal eletrônica de serviço, em geral os sistemas apresentam a implementação para a Cidade de São Paulo (cada município possui uma implementação própria);
- Oportunidade*: Projetos que podem ser iniciados mas ainda não foram comercialmente fechados;
- Orçamento: Geração de orçamento para clientes sem a obrigatoriedade de alteração em estoque ou no financeiro;
- Ordem de produção*: Também chamado de PCP controla a produção de produtos pela empresa; em geral transformando matérias-primas em produtos finais (ou intermediários);
- Ordem de serviço*: Controle de ordens de serviço que devem ser executadas ou que já foram encerradas;
- Produto: Gerenciamento de produtos, tanto nas suas características como valor de custo e venda;
- Projeto*: Gerenciamento de projetos que são (ou já foram) executados pela empresa e controle do seu status;
- Serviço: Gerenciamento de produto, tanto nas suas caracteristicas como valor de custo e venda;

- Tarefa*: Controle das tarefas que devem ser (ou já foram) executadas pelos funcionários ou terceiros, podem ou não estar associadas a um projeto;
- Transportadora: Gerenciamento de transportadoras que podem ser associadas as vendas e compras;
- Troca: Gerenciamento de trocas de produtos;
- Venda: Venda de produtos ou serviços com integrações com estoque (no caso de produto) e financeiro;
- WMS*: Gerenciamento de armazém desde a armazenagem passando pela inteligência de melhor posicionamento dos produtos até o envio para a transportadora.

Todas essas definições são genéricas e não necessariamente os sistemas analisados possuem todas essas funcionalidades integralmente; inclusive podem implementar outras funcionalidades.

Observa-se que uma característica determinante é a grande necessidade de integração entre os módulos principalmente para atualização de valores e garantia de integridade dos dados. Além disso a quantidade de informações que um sistema desse tipo deve processar e armazenar é grande e uma forma de facilitar a inclusão de dados pelo desenvolvedores garantiria um melhor aproveitamento de tempo uma vez que a atividade repetitiva fosse automatizada.

Durante a análise dos sistemas foi possível identificar um padrão na forma de organização e entrada de dados. Em geral existe uma listagem de todos os registros já existentes no sistema com opções de ações (como de edição, exclusão entre outros - dependendo do tipo de registro). A Figura 3.2 representa o *wireframe* deste padrão observado.

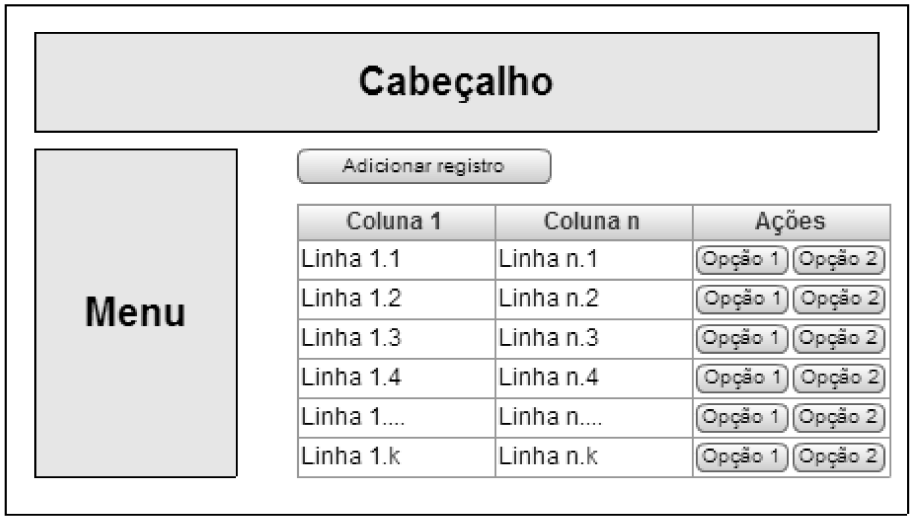

Figura 3.2: Wireframe do padrão de listagem de registros

Nessa listagem existe a opção também de inclusão de um novo registro, onde um formulário é aberto e, após a confirmação do seu preenchimento o dado é armazenado. A Figura 3.3 apresenta o *wireframe* deste padrão. Em geral o padrão para editar registro é o mesmo (trazendo os campos pré-preenchidos).

Figura 3.3: Wireframe do padrão para adicionar registro

É importante destacar que o *template* da interface da aplicação não foi observado e sim o padrão do *layout* e da navegação, ou seja, a arquitetura da informação. No *wireframe* o cabeçalho e o menu estão presentes apenas para uma melhor visualização já que não existe esse tipo de informação não é significativo para o atual estudo e não foi observado nas análises.

Apesar de alguns exemplos apresentados oferecerem opções como gerenciamento de serviços, PCP porém este trabalho se limitará ao escopo de sistemas ERPs voltados para varejo. Outras funcionalidades como emissão de boleto e geração de nota fiscal são muito específicas para cada caso não sendo considerado reúso portanto também excluídos desta pesquisa.

3.2.2 Mapeamento: Heurísticas de Nielsen x FUFs

Juristo, Moreno e Sanchez-Segura (2007a) compilaram uma única lista de trabalhos de diversos autores no campo de interação com o usuário. Os pontos desta lista foram nomeados FUFs, totalizando sete, cada um com mecanismos de usabilidade: FUFs são objetivos gerais que são especializados em mecanismos de usabilidade. Estes conceitos foram debatidos com detalhes na Seção 2.3.

Entre os autores estudados por Juristo, Moreno e Sanchez-Segura, Nielsen é citado diversas vezes e em alguns FUFs. Contudo não há uma relação explícita na maior parte dos casos.

Este mapeamento não foi realizado pelos autores e portanto, é proposto como uma contribuição desta dissertação. Como as heurísticas de Nielsen são amplamente conhecidas pelos designers de interação e os FUFs possuem maior foco no desenvolvimento funcional este mapeamento é útil para unir essas duas pontas.

Acredita-se ser relevante o mapeamento Heurísticas de Nielsen x FUF, pois isto auxiliaria a visão funcional e de interação pelos projetistas dos sistemas. Além disto, auxiliaria em uma possível inspeção de produto ou mesmo de um metamodelo.

O FUF1 possui foco na funcionalidade de *feedback* e é dividida em quatro mecanismos de usabilidade: (a) status do sistema, (b) interação, (c) alerta, (d) *feedback* de ações demoradas. A ideia geral deste FUF é que o usuário sempre esteja ciente sobre o que está acontecendo no sistema (a,d), saiba que suas ações foram devidamente recebidas (b) e seja alertado sobre as consequencias de suas ações (c).

É possível associar estes mecanismos de usabilidade às heurísticas H1, H5, H9 e H10. O FUF1a, FUF1b e FUF1d podem ser relacionados a H1, visibilidade do status do sistema, já que todos têm a base de informar o usuário sobre o que está acontecendo. FUF1c também se associa a H1, H5 (prevenção de erros), H9 (recuperação de erros) e H10 (ajuda e documentação) pois além de informar o usuário também evita que cometa erros, permite que desista da ação e o ajuda a seguir pelo caminho desejado.

Também dividida em quatro mecanismos de usabilidade, o FUF2 mantem o cerne em permitir que o usuário possa cancelar e desfazer determinadas ações; o mecanismos de usabilidade são (a) desfazer global, (b) desfazer de objeto específico, (c) abortar operações e (d) voltar. As duas primeiras são muito parecidas apenas com escopos diferentes (a primeira global e a segunda específica). A última entra neste mesmo grupo já que possui um escopo mais específico ainda (para uma única ação). O FUF2c participa do mesmo grupo porém seu escopo é para ações que estão sendo executadas e devem ser encerradas (é possível observar a continuação do FUF1d). Assim estas associam-se a H3 e H9 já que permitem que o usuário tenha mais controle sobre suas ações e possa se corrigir caso alguma ação tenha sido incorreta.

O FUF3, prevenção e correção de erros de preenchimento, possui apenas um mecanismo de usabilidade: entrada de texto estruturado. A ideia deste FUF é prover ao usuário ferramentas para não errar no momento do preenchimento dos dados e, caso erre, possa se corrigir facilmente. Assim as heurísticas H4 (consistência e padrões), H5, H6 (reconhecimento ao invés de lembrança) e H9 podem ser citadas como associações válidas; a primeira pois os padrões fazem com que o preenchimento dos dados seja mais simples, a terceira pois auxilia o usuário a lembrar qual o formato adequado de entrada e a segunda e última visam prevenir e corrigir erros.

O FUF4, *wizard*, também com apenas um mecanismo de usabilidade (execução passo-a-passo) possui o foco em ajudar o usuário de forma interativa a realizar alguma ação. Desta forma a heurística H10 pode ser associda.

Possuindo três mecanismos de usabilidade, o FUF5, tem foco no perfil do usuário: (a) salvar funções preferenciais do usuário, (b) salvar interfaces preferenciais do usuário e (c) salvar favoritos do usuário. Assim como o FUF2 neste os dois primeiros mecanismos de usabilidade são muito parecidos apenas com escopos diferentes, ambos são combinados com H7 (flexibilidade e efeciência no uso) já que sua aplicação torna as atividades do usuário mais simples e corriqueiras. Já o FUF5c associa-se a H6 uma vez que o usuário não precisa se recordar onde está cada função que utiliza, elas estão em um único local salvas.

O FUF seguinte, FUF6, com apenas um mecanismo de usabilidade, define que deve ser disponibilizado ajuda em vários níveis para o usuário. Temos as seguintes heurísticas associdas: H2 (ligação entre sistema e o mundo real), H6, H8 (estética e design minimalista) e H10. Na primeira uma vez que as informações estão claras para o usuário não fica confuso ao usar o sistema; a ajuda provisionada na heurística seguinte permite que o usuário possa visualizar facilmente

dados informados outrora para o sistema; a partir do momento que o sistema possuir um design limpo e que não confunda o usuário não precisará de ajudas mais específicas (e complicadas) - H8. Por fim a documentação do sistema (H10) é mais um nível de ajuda necessário e presente nos sistemas.

O último, FUF7, possui um mecanismo de usabilidade: agregação de comandos. Usuários mais experientes podem usar "atalhos"para realizar algumas atividades de forma mais rápida; portanto associa-se a heurística H7.

A Tabela 3.6 apresenta, de forma resumida, a relação entre Heurísticas de Nielsen e FUFs. As relações são indicadas por um X na célula de cruzamento entre a heurística e o FUF.

Tabela 3.6: Associção entre Heurísticas de Nielsen e FUFs

	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10
FUF1a	X									
FUF1b	X									
FUF1c	X				X				X	X
FUF1d	X									
FUF2a			X						X	
FUF2b			X						X	
FUF2c			X						X	
FUF2d			X						X	
FUF3				X	X	X			X	
FUF4										X
FUF5a							X			
FUF5b							X			
FUF5c						X				
FUF6		X				X		X		X
FUF7							X			

Este trabalho, a partir deste ponto, usará Heurísticas de Nielsen, FUFs e sua relação para embasar as decisões feitas e resultados obtidos.

3.3 MetaUsaERPWeb

Apesar do UML ser adotado como padrão pela OMG alguns autores defendem que por ser tão abrangente se torna difícil ser utilizado por não-especialistas. Ao mesmo tempo não é tão fácil adaptá-lo para necessidades específicas de domínio. Para solucionar este tipo de situação existe a DSM que foca em um grupo específico de domínios apresentando soluções mais adequadas para o grupo em questão.

Assim, aplicando o DSM juntamente com transformações M2C no desenvolvimento de sistemas ERP e pode-se produzir uma grande quantidade de código com seus módulos e relações é gerado, necessitando pouca ou nenhuma adaptação dependendo do objetivo do sistema ERP.

A etapa de engenharia de domínio contou com a investigação do domínio de sistemas ERP, exposta na Seção 3.2.1, para preparação e organização dos requisitos mínimos que devem ser

atendidos por um sistema ERP. Esses requisitos foram utilizados para desenvolvimento dos artefatos reutilizáveis, no caso deste trabalho, o metamodelo e seu transformador. Os próximos passos seguidos na etapa de engenharia de domínio serão abordados nas próximas subseções, além disto é apresentado a verificação da proposta.

3.3.1 Modelagem Específica de ERP de varejo

Considerando as motivações e desafios no desenvolvimento de sistemas ERPs de varejo com elementos de usabilidade, este trabalho propõe um metamodelo, denominado *MetaUsaERPWeb*, com o intuito de facilitar o desenvolvimento desse tipo de aplicação automatizando pontos específicos do domínio de ERPs e aplicando conceitos de usabilidade. Este trabalho também contemplou o desenvolvimento de um transformador *model-to-code* do metamodelo proposto e geração das tabelas do banco de dados.

Observou-se que o domínio de sistemas ERPs é muito abrangente para ser abordado de forma integral por este trabalho. Como não existe um conceito único de funcionalidades de um sistema ERP já que este deve se adaptar as necessidades de quem está atendendo, a contribuição desse trabalho se focou no cenário específico de mercado de varejo onde existem fortes ligações entre as áreas da empresa, principalmente venda e estoque. Futuramente os conceitos aplicados nesse trabalho podem ser ampliados a fim de possibilitar um maior alcance de cenários.

Também constatou-se que a abordagem de todos os pontos das Heurísticas de Nielsen não poderia ser realizada de forma integral por este trabalho. Assim considerou-se os pontos mais relevantes e recorrentes a partir da análise heurística apresentada na Seção 3.2.1. Estes pontos foram considerados baseando-se na forma que a funcionalidade poderia ser automatizada facilitando o desenvolvimento e no impacto que sua implementação causaria na interação do usuário (segundo a Tabela 3.1). Dessa forma as seguintes Heurísticas de Nielsen foram trabalhadas nesse projeto:

- H1: visibilidade do status do sistema;
- H4: consistência e padrões;
- H5: prevenção de erros;
- H6: reconhecimento ao invés de lembrança;
- H9: recuperação de erros.

Considerando-se os FUFs podemos citar:

- FUF1: *feedback*;
- FUF3: prevenção e correção de erros de preenchimento.

Segundo a Tabela 3.6 a H9 também relaciona-se com o FUF2, porém este não foi considerado na primeira versão deste trabalho devido a sua complexidade de implementação ficando como uma extensão do MetaUsaERPWeb.

O desenvolvimento deste trabalho seguiu por três fases: metamodelagem das funcionalidades referentes ao ERP, metamodelagem das funcionalidades referentes as características de usabilidade e desenvolvimento do transformador M2C (*model-to-code*).

Para implementação do metamodelo utilizou-se o *plugin* EMF[13] (*Eclipse Modeling Framework Project* da IDE Eclipse[14]. Já para desenvolvimento do transformador fez-se uso do *plugin* JET[15], (*Java Emitter Template*) também da IDE Eclipse, baseado em *templates*.

Metamodelagem das funcionalidades

Durante o desenvolvimento do metamodelo, sob ponto de vista dos sistemas ERP, foi colocado em foco a integração entre módulos (atualização de valores e garantia de integridade de dados) e operação de CRUD[16]. A partir destes elementos é possível criar regras de consistência de valores, mapeamento de campos necessários, geração de interface e criação de estrutura de banco de dados.

Metamodelagem das características funcionais de usabilidade

Considerando-se o exposto na subseção 3.2.1 alguns pontos práticos foram levados em consideração no desenvolvimento do metamodelo como definição de tipos de dados com a possibilidade de indicar ao usuário a forma correta de preenchimento e permitir a validação do campo preenchido a fim de evitar erros (FUF3). Ainda no objetivo de guiar o usuário durante o preenchimento dos dados a inclusão de textos de ajuda tem importância no processo e são considerados no metamodelo.

A incorporação das características de usabilidade no modelo proposto foi dividida em quatro etapas (PANACH et al., 2015):

1. Identificar MoUs para cada mecanismo de usabilidade;
2. Identificar propriedades que configuram cada MoU;
3. Definir primitivas conceituais para representar de forma abstrata as propriedades do modo de uso;
4. Descrever as mudanças que precisam ser executadas no transformador a fim de implementar as propridades identificadas.

Panach et al. (2015) apresentaram estes passos para serem seguidos no caso de um metamodelo já existente não possuir suporte a questões de usabilidade, portanto execução de uma reengenharia de metamodelo. No caso do trabalho apresentado nesta dissertação o metamodelo já está sendo concebido com esta característica; de qualquer forma a proposta dos autores também pode ser utilizada uma vez que apresenta uma metodologia de inclusão de funcionalidades em metamodelos genéricos.

[13]https://www.eclipse.org/modeling/emf/
[14]https://www.eclipse.org/
[15]http://www.eclipse.org/emft/projects/jet/
[16]Acrônimo para *Create*, *Read*, *Update* e *Delete*, ou seja, operações básicas sobre informações de criar, ler, atualizar e deletar

Entretando a partir das análises apresentadas na subseção 3.2.1 e do exposto nas seções anteriores identificou-se que sistemas ERP possuem uma característica funcional que impacta na usabilidade: a consistência das informações. Junto com a integração entre os módulos também uma série de regras podem ser aplicadas a fim de garantir que as regras de negócio do sistema sejam respeitadas e, como consequencia, que as informações se mantenham consistentes. Não foi identificado nenhum FUF com essa característica. Portanto este trabalho propõe um novo FUF com essa intenção estentendo o trabalho de Juristo, Moreno e Sanchez-Segura (2007b).

Este FUF, denominado *consistência de informações* (FUF8), possui dois UMs:

(a) Alerta de inconsistência: informar o usuário que sua ação tornará algumas informações inconsistentes de acordo com a regra de negócio definida;

(b) Condução no preenchimento: auxiliar o usuário a preencher o dado corretamente visando manter a consistência de acordo com a regra de negócio.

Um cenário possível para exemplificar este FUF e seus UMs é de uma empresa de revenda de produtos a pronta entrega, ou seja, a venda só pode ser realizada caso o produto esteja em estoque. Em seu ERP existe, ao menos, uma regra: o estoque não pode tornar-se negativo. Assim caso alguma ação possa resultar em tornar o estoque negativo, ou seja, retirar uma quantidade de produto maior que exista em estoque, deve ser interrompida. Assim o usuário deve ser avisado desta regra e que sua ação irá tornar as informações inconsistentes, está é a função do primeiro UM. Nesta mesma ação o usuário poderá ser informado qual é o maior valor possível para ser retirado do estoque, contemplando assim o segundo UM.

Durante a análise funcional da Subseção 3.2.1 foi identificada uma funcionalidade de usabilidade que pode ser utilizada para o UM.b do FUF8: o autocompletar. O usuário pode receber sugestões que o auxilie a manter a consistência das informações. No caso da Figura 3.1 o preço é autocompletado auxiliando o usuário a saber qual o valor padrão do produto e, a partir dele, terá mais conhecimento e base para, por exemplo, oferecer um desconto ao cliente.

O FUF8 aqui recomendado foi considerado no desenvolvimento do metamodelo e do transformador propostos neste trabalho.

3.3.2 Desenvolvimento do metamodelo

Uma vez definido os pontos que o metamodelo deve atender iniciou-se o seu desenvolvimento. A Figura 3.4 apresenta o metamodelo completo e os FUFs relacionados (a relação entre FUF e heurística foi feita de acordo com o mapeamento na Tabela 3.6). O FUF1 é implementado pelo transformador, com isso não possui entidades no metamodelo. Este é composto por seis componentes, o central é o *ERP* que possui *entities* e *data types*. *Entities* são compostas por *attributes* e outras *entities* através das *entities related*.

O componente *ERP* é utilizado para organizar, associar e armazenar todos os outros componentes. O componente *data type* é responsável por definir tipos de dados que serão disponibilizados no sistema, possui três atributos: *name*, que define um identificador para o *data type*; *helperText*, que especifica o texto de ajuda para preenchimento deste dado; e *regex*, que define a expressão regular validadora deste dado e pode definir também máscaras de campos.

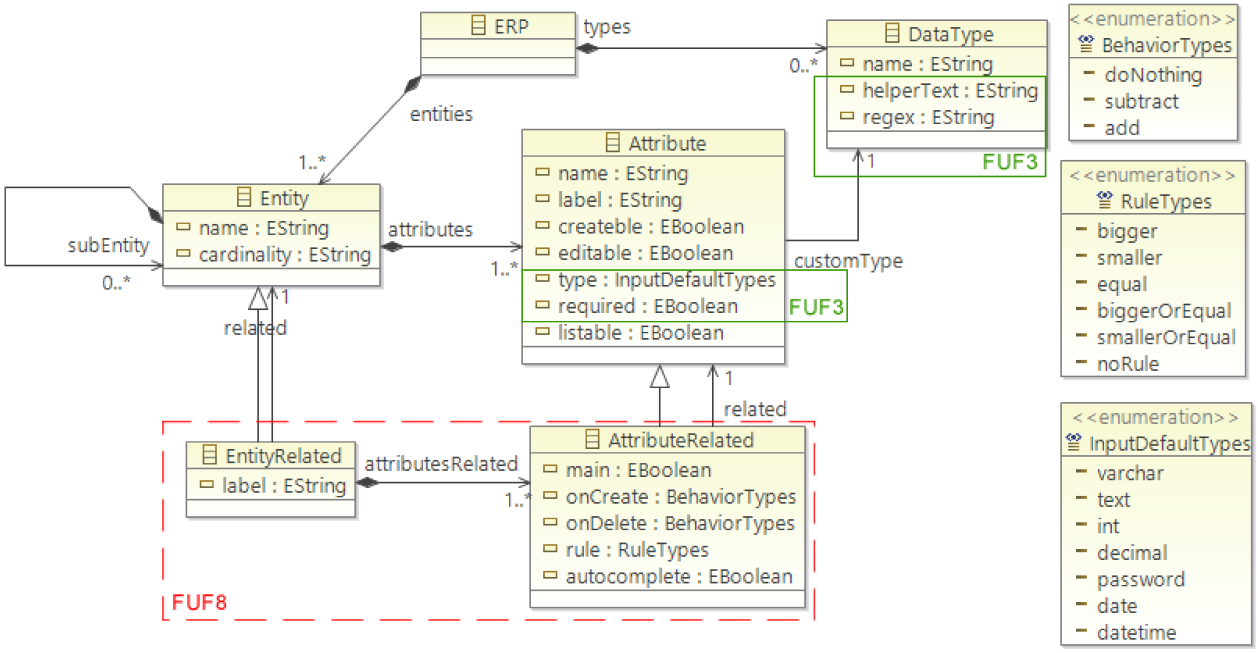

Figura 3.4: Metamodelo completo

O *data type* possui grande foco na questão de usabilidade já que auxilia o usuário no preenchimento correto dos dados. As heurísticas H5, H6 e H9 são parcialmente ou integralmente satisfeitas por este componente. O atributo *helperText* auxilia o usuário a como preencher o dado no momento inicial de entrada (H5 e H9) e caso tenha algum problema na validação (H9). Já o atributo *regex* previne os erros (H5) e auxilia o usuário a lembrar como preencher os dados (H6) no caso de máscaras de campos; a recuperação de erros (H9) é feita através da validação do campo pela expressão regular. O FUF3 (prevenção e correção de erros de preenchimento) é satisfeito.

O enumerador *InputDefaultTypes* define padrões de *data types*. Esses padrões foram considerados os mais comuns encontrados em sistemas ERPs, dessa forma o desenvolvedor não necessita definí-los manualmente através do componente *data type*. Os tipos definidos nesse numerador possuem tratamento especial pelo transformador. Existem sete *data types* padrões:

- *varchar*: campo de texto de até 255 caracteres;
- *text*: campo de texto com mais de 255 caracteres;
- *int*: número inteiro;
- *decimal*: número real;
- *password*: senha;
- *date*: data;
- *datetime*: data e hora.

O componente principal do metamodelo é a *entity*, este representa cada módulo do ERP; uma *entity* pode possuir várias *sub-entities*, que também são *entities*, e estas outras. Cada *entity* possui dois atributos: *name*, que define um identificador para a *entity*; *cardinality* define a cardinalidade máxima da *sub-entity*, ou seja, quantas daquela *sub-entity* são aceitas na *entity* pai. O atributo *cardinality* deve ser preenchida com um número, que representa o número máximo de *sub-entities*, ou com *, representando que não existe limite máximo. A cardinalidade mínima é zero.

A *entity* possui um ou mais *attributes*. Cada *attribute* define um dado que compoem a *entity* e possui sete atributos: *name*, que define um nome para o *attribute*; *label*, rótulo do campo para exibição para o usuário; *createble*, *editable* e *listable* define se o campo pode ser exibido e gerenciado nas telas de criação e edição e se o campo faz parte da tabela na listagem de registros, esses atributos são consideram o conceito de CRUD. O atributo *required* define se o campo é de preenchimento obrigatório. Além do atributo *type* também existe a relação com o componente *data type*, apenas um necessário, ele define o tipo do campo que o *attribute* corresponde.

O componente *attribute related* é derivado do componente *attribute*, ou seja, possui todos os seus atributos. O *attribute related* é peça chave para o funcionamento do relacionamento dos módulos conforme identificado na subseção 3.2.1. Este componente se relaciona a uma outra *entity* através da *entity related* e ao *attribute* desta *entity*. Este relacionamento é controlado através dos atributos do *attribute related*: *onCreate* e *onDelete* definem qual o comportamento deste *attribute related* ao ser criado ou excluído. Existem três comportamentos possíveis: adicionar (*add*), subtrair (*subtract*) ou não fazer nada (*doNothing*); assim ao criar ou excluir o *attribute related* o *attribute* relacionado sofrerá o comportamento definido; na edição deve ser executado o comportamento da exclusão e então da criação. O atributo *rule* define qual regra de negócio deve ser aplicada sobre a relação, existem seis regras:

- *bigger*: o *attribute related* deve ser maior que o *attribute* relacionado;
- *smaller*: o *attribute related* deve ser menor que o *attribute* relacionado;
- *equal*: o *attribute related* deve ser igual ao *attribute* relacionado;
- *biggerOrEqual*: o *attribute related* deve ser maior ou igual que o *attribute* relacionado;
- *smallerOrEqual*: o *attribute related* deve ser menor ou igual que o *attribute* relacionado;
- *noRule*: não existe regra para essa relação.

Essas regras também são observadas na questão de usabilidade do sistema já que podem ser usadas a fim de auxiliar na prevenção de erros e recuperação de erros.

Os atributos *main* e *autocomplete* estão relacionados a fim de prover a funcionalidade de autocompletar na subseção 3.2.1. O primeiro atributo define se aquele *attribute related* é o principal; já o segundo define que, quando o *attribute related* principal for selecionado então este deve ser autocompletado.

A *entity related* possui apenas um atributo: *label* que define um nome amigável para a relação ser exibida ao usuário.

A dupla *entity related* e *attribute related* definem algumas regras de negócio que são necessárias para o transformador satisfazer o FUF8.

Através do metamodelo a heurística H4 é satisfeita já que os nomes e termos são definidos apenas uma vez ficando a cargo das relações (e do transformador) transportar estes nos locais adequados. Já a heurística H1 e o FUF1 são satisfeitos pelo transformador.

O EMF, *plugin* utilizado para a metamodelagem, também possui modo de modelagem (utilizando o metamodelo desenvolvido). Esse modelo é persistido na forma de um arquivo XML, denominado XMI[17] (*XML Metadata Interchange*), podendo ser utilizado como entrada de tranformadores, inclusive no que será abordado na próxima subseção.

3.3.3 Desenvolvimento do transformador

O artefato resultante dos esforços da subseção anterior é um PIM, ou seja, um modelo independente de plataforma podendo ser implementado por qualquer linguagem de programação. Entretanto para o desenvolvimento do transformador M2C foi necessário escolher uma plataforma e linguagem de programação como artefato alvo.

Foi escolhida a plataforma Web, com o código servidor executado pela linguagem PHP[18]. O código cliente gerado em HTML e Javascript e o banco de dados MySQL[19]. O código gerado segue o padrão arquitetural MVC (*Model-View-Controller*) (WANG, 2011) e utiliza o framework CakePHP[20] em sua versão 2.6.2.

O transformador possui grande preocupação na qualidade da interface seguindo regras definidas pelos FUFs e Heurísticas de Nielsen. Apesar disso permite alteração completa do *template* da interface da aplicação onde o desenvolvedor pode alterar completamente estilos e estrutura das páginas configurando os arquivos HTML, CSS e Javascript de acordo com o funcionamento do transformador, respeitando padrões e obrigatoriedades definidos. Uma vez realizada a configuração o transformador gera os arquivos finais em HTML, respeitando as definições anteriormente realizadas, e mantendo os arquivos CSS e Javascript. Existem funcionalidades implementadas em Javascript que são automaticamente incluidas principalmente com o objetivo de satisfazer as necessidades de usabilidade impostas pelo metamodelo e pelo modelo.

Quanto a transformação das funcionalidades descritas na Subseção 3.3.2 as propriedades não configuráveis são implementadas pelo transformador sem a necessidade de conferência no modelo em questão. Já para as propriedades configuráveis o transformador verifica a decisão presente no modelo (PANACH et al., 2015). Dessa forma, por exemplo, a possiblidade de apresentar *feedback* ao usuário é implementado automaticamente pelo transformador sem a necessidade de indicação formal do analista no momento de criação do modelo ficando apenas ao seu critério a definição de onde a mensagem será exibida para o usuário no *layout* da interface.

O JET, *plugin* utilizado para a implementação do transformador, é baseado em *templates*,

[17]Padrão da OMG para troca de informações
[18]http://www.php.net/
[19]http://www.mysql.com/
[20]http://cakephp.org/

ou seja, o código gerado pelo transformador possui um modelo baseado no metamodelo onde existem variáveis que tem o valor preenchido pelo modelo. A Figura 3.5 apresenta um exemplo de *template* e sua ligação com o metamodelo e modelo. O código apresentado é o *template*, nesse caso apenas o trecho <%= *entityName* %> (destacado em amarelo) é variável. Dessa forma o tranformador, através do metamodelo, tem conhecimento que o componente *Entity* possui um atributo *name* (destacado em vermelho) que é uma *string*.

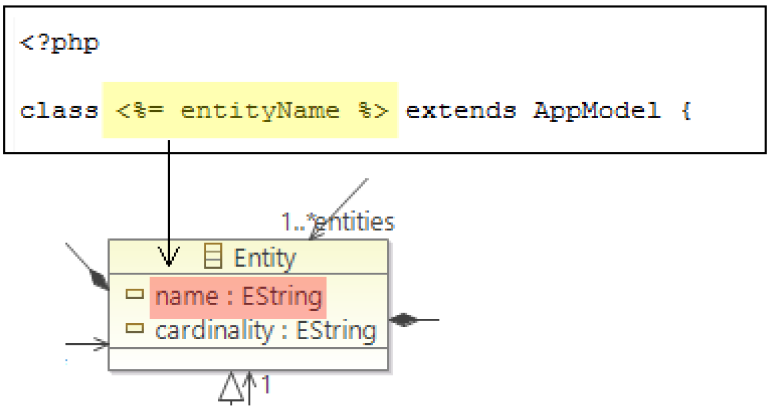

Figura 3.5: Exemplo de *template* do JET

O transformador proposto neste trabalho, além das informações do modelo baseado no MetaUsaERPWeb, possui também uma série de configurações de saída de arquivos e de *layout* da aplicação. Essas configurações são definidas na pasta *configs* do projeto do transformador. As configurações de saída são definidas no arquivo *general.xml*. Esse arquivo possui dois parâmetros: *rootDirectory* e *metaErpModelPath*; o primeiro define qual é a pasta raiz que os arquivos devem ser criados, onde é importante destacar que nessa pasta já devem existir os arquivos do CakePHP (versão 2.6.2), ou seja, o transformador não cria os arquivos padrões; já o segundo define qual o local que o modelo a ser transformado está. Essas duas informações estão em um nó pai, *generalConfigs*, a Figura 3.6 apresenta um exemplo do conteúdo deste arquivo.

```
<generalConfigs>
    <rootDirectory>PastaRaiz</rootDirectory>
    <metaErpModelPath>LocalDoModelo</metaErpModelPath>
</generalConfigs>
```

Figura 3.6: Exemplo do arquivo *general.xml*

A Figura 3.7 apresenta a organização interna do transformador; existem três componentes: *template*, *generators* e *loaders*. O primeiro contém o template que será completado gerando o arquivo final após execução do transformador (conforme exemplo na Figura 3.5), o segundo recebe as informações do XMI do modelo gerado através do EMF, por fim o último carrega as configurações de HTML para criar o template de acordo com o configurado pelo desenvolvedor a partir das configurações de *layout*. Existem outros *templates*, *generators* e *loaders* no

transformador porém esses são os principais e recebem informações desses outros.

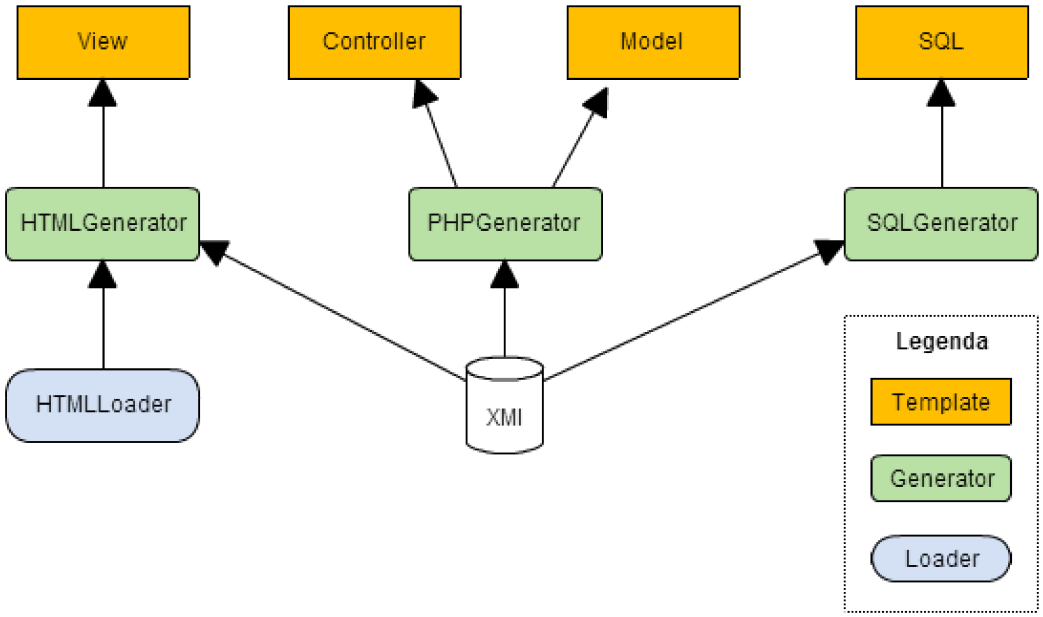

Figura 3.7: Diagrama do transformador

Os *loaders* possuem uma série de regras para configuração e personalização do *layout* da aplicação e são responsáveis por informar o *HTMLGenerator* qual é o *layout* final da aplicação. Essas regras e configurações são definidas na pasta *html*, dentro de *configs*.

Existem cinco grupos de configurações e mais uma configuração avulsa; cada grupo possui uma série de configurações referente ao seu grupo onde cada grupo é, em geral, uma tela específica gerada automaticamente pelo transformador. Todas as configurações de grupos são arquivos HTML e caso não sejam especificados serão ignorados pelo transformador e nada será exibido.

O primeiro grupo, *allPages*, define oito configurações primárias que afetam todas as páginas da aplicação:

- *beforeAllContentPages*: define o conteúdo HTML/Javascipt que será renderizado primeiro (antes de qualquer outro conteúdo); em geral, define-se o cabeçalho e menu;
- *afterAllContentPages*: define o conteúdo HTML/Javascipt que será renderizado por último (depois de todos os outros conteúdos); em geral, define-se o rodapé;
- *beforeFlash*: *flash* é a mensagem de *feedback*, sempre exibida na parte de cima do *layout* logo após o conteúdo definido pelo *beforeAllContentPages*. Essa configuração define, por exemplo, a *box* que a mensagem será exibida;
- *afterFlash*: define o que será exibido após a mensagem, em geral, fecha o *box* aberto pela configuração *beforeFlash*;
- *beforeFlashMessage*: define o que será exibido imediatamente antes da mensagem de *flash* como um ícone por exemplo;

- *afterFlashMessage*: define o que será exibido imediatamente após a mensagem de *flash* como o comando para o usuário esconder a mensagem como um [X];
- *beforeTitle*: define o que será renderizado antes do título da página, em geral, a configuração de tamanho de texto, cores, etc;
- *afterTitle*: define o que será renderizado após o título da página.

A Figura 3.8 apresenta o *wireframe* dessas configurações. Os *boxes* com fundo cinza representam as configurações, com fundo branco representam dados que serão populados pelo transformador ou em tempo de execução pela aplicação e com fundo amarelo representa o espaço que será utilizado em outros momentos pelo transformador e possui suas configurações próprias.

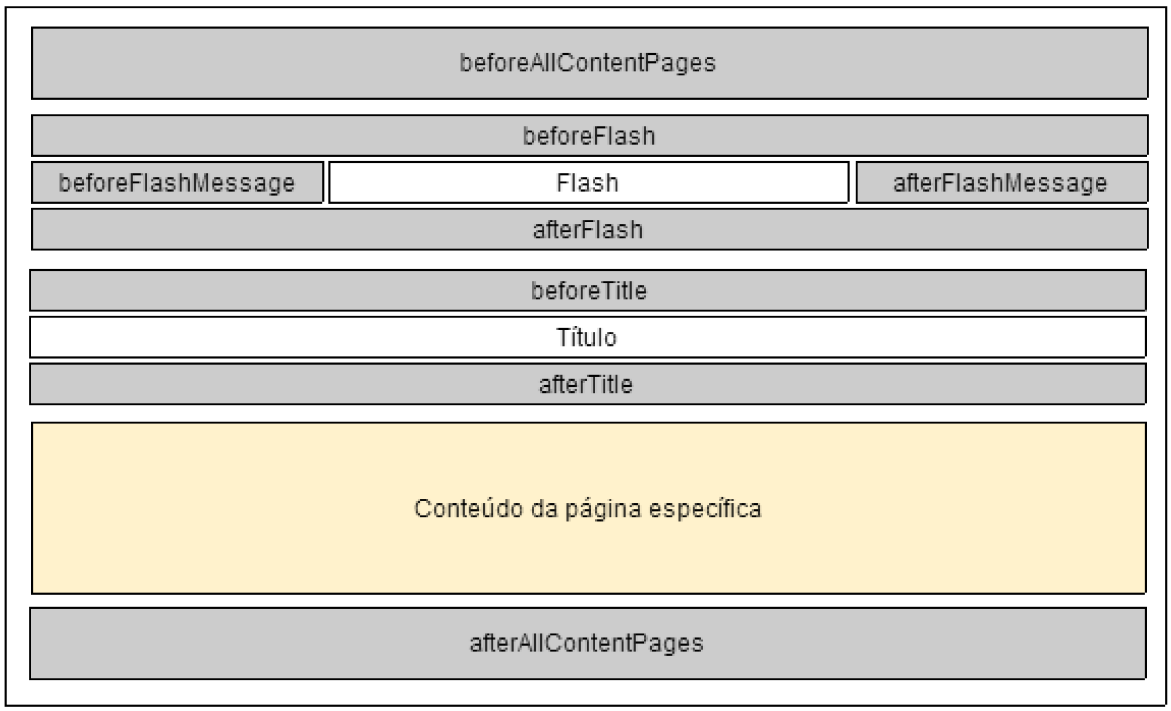

Figura 3.8: *Wireframe* do grupo *allPages*

É importante destacar que a configuração *beforeAllContentPages* é responsável pela exibição do menu. O local do menu é presentado pela marcação de texto {*MENU_LIST*} e o transformador se resposabilizará por exibir o menu nesse espaço.

O menu também possui configurações: as configurações secundárias do grupo *allPages*. O menu é codificado como uma lista e suas configurações podem estilizá-lo, estas configurações são classes CSS. Existem quatro configurações secundárias:

- *ulMenuExtra*: define classes e atributos para o elemento *ul* da lista;
- *liMenuExtra*: define classes e atributos para os itens *li* da lista;
- *beforeMenuItem*: define o que será exibido imediatamente antes do texto do item do menu;

- *afterMenuItem*: define o que será exibido imediatamente após do texto do item do menu.

O menu representa cada *Entity* pai e cada item do menu possui um link para a página principal do módulo. A Figura 3.9 apresenta o template do menu, as configurações estão entre colchetes.

```
<ul{ulMenuExtra}>
    <li{liMenuExtra}><a href="Link"/>{beforeMenuItem}Módulo{afterMenuItem}</a></li>
</ul>
```

Figura 3.9: *Template* do menu com suas configurações

Em geral os *feedbacks* possuem padrão visual onde avisos de erros são exibidos em vermelho, de sucesso em verde e de informações gerais em azul. Assim é possível configurar uma classe CSS para exibir a cor correta (ou outros elementos visuais). Isso é feito através do arquivo *flash.xml* localizado na pasta *html*. Este arquivo possui três configurações: *successClass*, *errorClass* e *infoClass* que representam, respectivamente, as classes de sucesso, erro e informações gerais. Essas informações estão localizadas em um nó pai, *flashConfigs*.

A Figura 3.10 apresenta um exemplo do arquivo *flash.xml* onde a classe de sucesso é *alert-success*, de erro é *alert-danger* e de informações gerais é *alert-info*.

```
<flashConfigs>
    <sucessClass>alert-success</sucessClass>
    <errorClass>alert-danger</errorClass>
    <infoClass>alert-info</infoClass>
</flashConfigs>
```

Figura 3.10: Exemplo de preenchimento do arquivo *flash.xml*

O segundo grupo, *create*, determina as configurações do *layout* das páginas que renderizam o formulário para a inclusão de informações no sistema. Esse grupo não define as configurações do formulário, apenas da página em si. O escopo desta página está localizado no *box* de fundo amarelo da Figura 3.8. Existem apenas duas configurações para esse grupo:

- *beforeThisContentPage*: define tudo que será renderizado antes de qualquer outro elemento desta página específica;
- *afterThisContentPage*: define tudo que será exibido após de qualquer outro elemento desta página específica.

O terceiro grupo, *edit*, também possui esse mesmo padrão. Este grupo, porém, define as configurações das páginas que exibem o formulário para edição das informações no sistema. A Figura 3.11 representa o *wireframe* destas configurações.

O quarto grupo, *form*, define as configurações de *layout* dos formulários presentes nas páginas de criação e edição das informações. E esse grupo define as regras que são aplicadas no *box* de fundo amarelo da Figura 3.11. Este grupo possui 16 configurações:

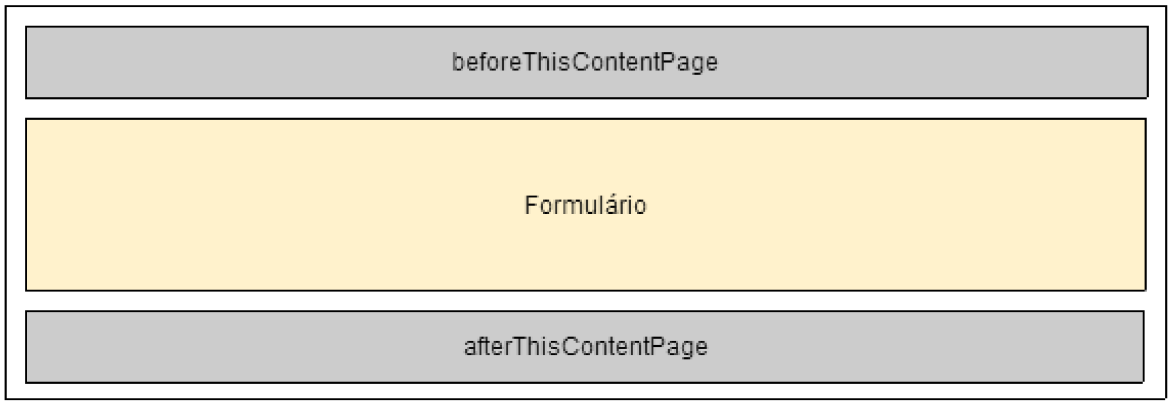

Figura 3.11: *Wireframe* dos grupos *create* e *edit*

- *beforeFormTag*: define o que será exibido antes da abertura da *tag* HTML *form*;
- *afterFormTag*: define o que será renderizado após o fechamento da *tag* HTML *form*;
- *formTagExtra*: define classes ou atributos extras para a *tag form*;
- *openBox*: define a abertura do *box* de *entities* relacionadas. Se a *entity* pai possuir *entities* filhas para melhor organização das informações elas podem ser segmentadas de forma visual em *entities* através de *boxes*;
- *closeBox*: define o fechamento do *box* de *entities* relacionadas;
- *beforeLabelTag*: define o que será apresentado antes do rótulo de cada item do formulário;
- *afterLabelTag*: define o que será exibido após o rótulo de cada item do formulário;
- *labelTagExtra*: define classes ou atributos extras para a *tag label*;
- *beforeInputTag*: define o que será renderizado antes da *tag input* de cada item do formulário;
- *afterInputTag*: define o que será exibido após a *tag input* de cada item do formulário;
- *labelInputExtra*: define classes ou atributos extras para a *tag input*;
- *beforeHelpText*: define o que será exibido antes do texto de ajuda de cada item do formulário;
- *afterHelpText*: define o que será apresentado após a *tag input* de cada item do formulário;
- *beforeSubmitTag*: define o que será renderizado antes do botão de enviar formulário;
- *afterSubmitTag*: define o que será exibido após o botão de enviar formulário;
- *labelInputExtra*: define classes ou atributos extras para a *tag* de *submit*.

A Figura 3.12 representa as configurações para o grupo *form*. Nela existem *boxes* de fundo verde que representam dados que são populados pelo transformador mas possuem alterações causadas pelas configurações, estas são apresentadas entre colchetes. Os elementos indicados no agrupamento 1 indicam que serão repetidos para quantas *entities* existirem associadas a *entity* pai (a *entity* pai também é considerada na exibição). Já os elementos indicados no agrupamento 2 representam o conjunto que é repetido com base na quantidade de atributos da *entity* permitidos para essa operação verificando-se *isCreateble* e *isEditable*.

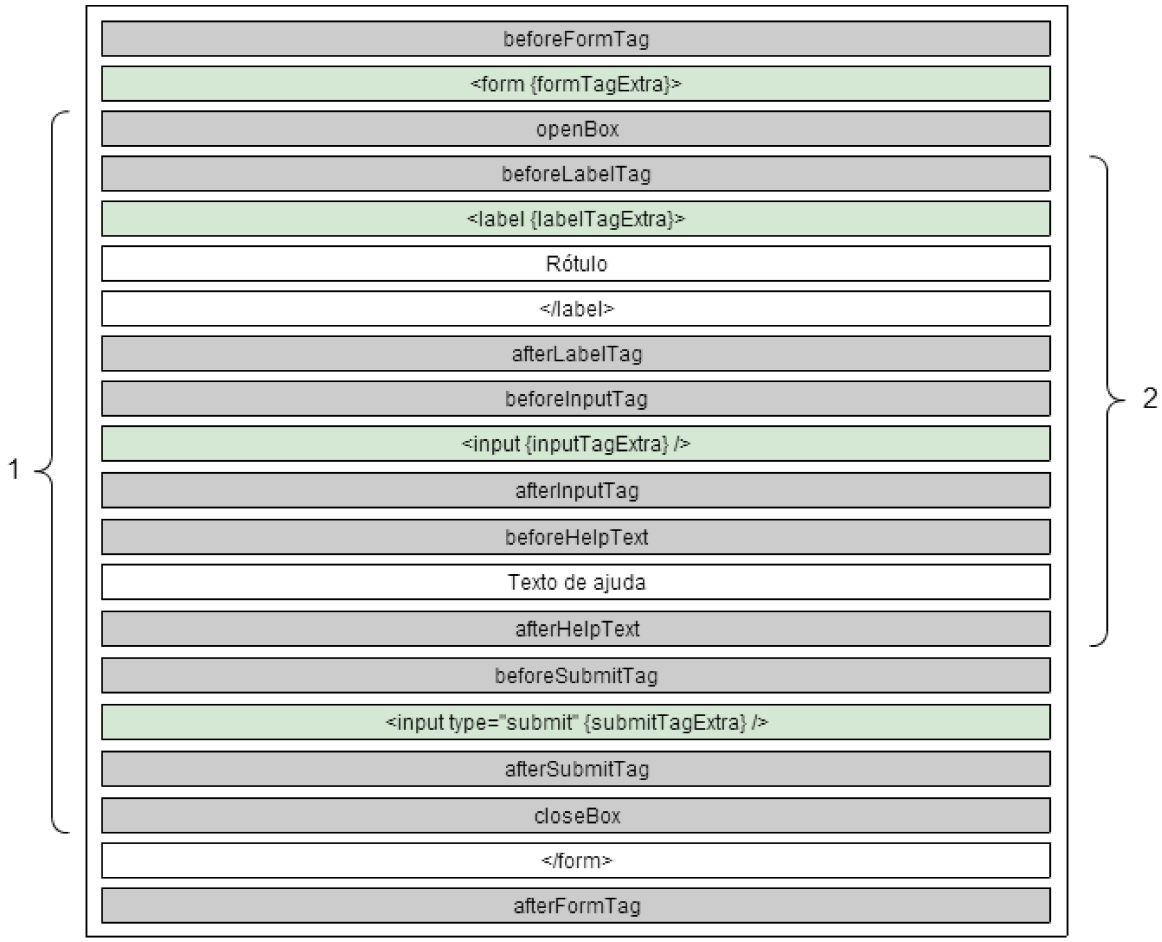

Figura 3.12: *Wireframe* do grupo *form*

O último grupo, *listingTable*, define as configurações usadas na página de listagem de registros já existentes no sistema. Esta página está localizada no *box* de fundo amarelo da Figura 3.8. Existem seis configurações para esse grupo:

- *beforeThisContentPage*: define tudo que será renderizado antes de qualquer outro elemento desta página específica;
- *afterThisContentPage*: define tudo que será exibido após de qualquer outro elemento desta página específica;
- *createButtonExtra*: define classes e atributos extras de botões desta página e de páginas pertencentes aos grupos *create* e *edit*;

- *beforeListingTable*: define o que será renderizado antes da tabela de registros
- *afterListingTable*: define o que será apresentado após a tabela de registros;
- *listingTableExtra*: define classes ou atributos extras para a *tag table*.

A Figura 3.13 apresenta o *wireframe* de configurações disponíveis do grupo *listingTable*.

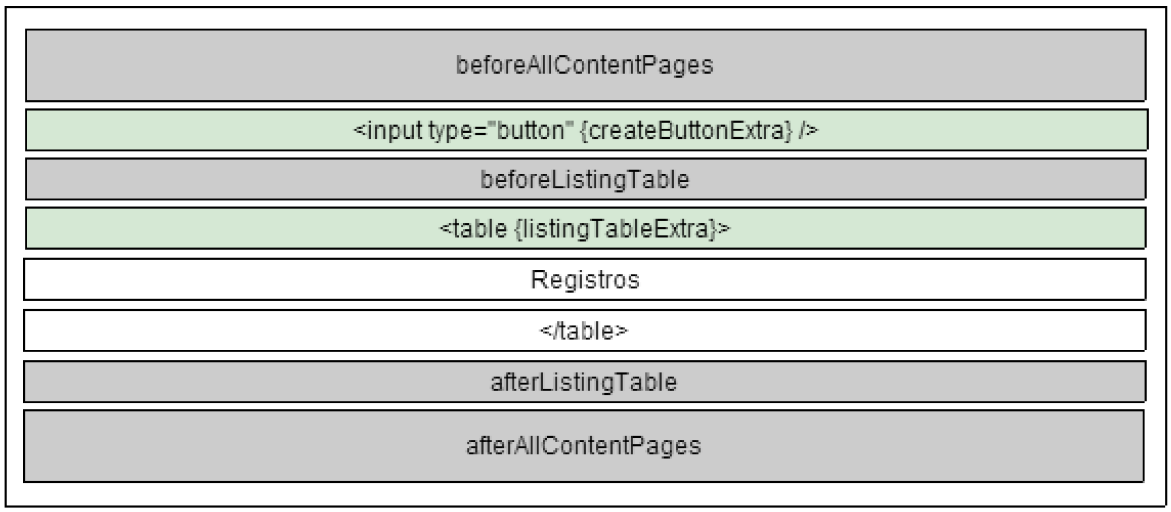

Figura 3.13: *Wireframe* do grupo *listingTable*

Além das configurações outra entrada necessária para a geração dos arquivos codificados é o modelo em formato XMI do MetaUsaERPWeb. Este arquivo define quais são as especificações da aplicação definida nos moldes do metamodelo.

A Figura 3.14 apresenta um exemplo de tranformação através de templates conforme o *plugin* JET. A primeira *box* representa o *template*, estão marcados em vermelho as configurações que são substituidas de acordo com as regras de *layout* da aplicação apresentadas anteriormente. Em amarelo estão marcados os espaços que são preenchidos de acordo com as informações do modelo. Este exemplo foi retirado transformador. Existem três substituições: nome do atributo, rótulo do atributo e indicação de obrigatoriedade (se existir a obrigatoriedade um asterisco é exibido, caso contrário nada é apresentado).

O XMI, segundo *box*, apresenta a *entity product* com quatro *attributes*: nome, preço, quantidade e descrição; sendo que os três primeiros são obrigatórios. É possível observar três marcações em amarelo nos três primeiros *attributes* demonstrando o valor de nome do produto, rótulo do produto e sua obrigatoriedade; no último *attribute* percebe-se que existem apenas duas marcações, como não é obrigatório então sua marcação é suprimida.

Finalmente a última *box* demonstra o código HTML baseado no *template* com as informações do XMI. As setas vermelha, verde e azul apontam o fluxo da transformação para o *attribute ProductName*; os outros são tratados igualmente.

A Figura 3.15 apresenta o trecho de código do transformador responsável pela apresenção do *helper text* dos atributos do formulário. No trecho é possível verificar que ocorre a renderização das configurações *beforeHelpText* e *afterHelpText*; entre estas o texto é exibido após uma série de verificações e tratamentos implementados pela classe *InputDefaultTypesHelper*.

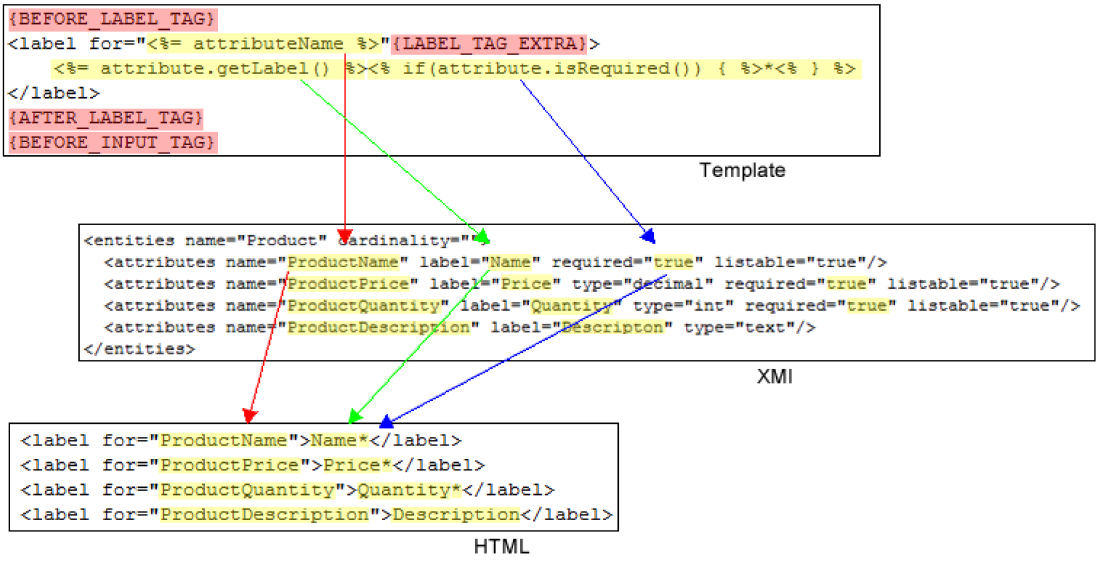

Figura 3.14: Exemplo de transformação

```
{BEFORE_HELP_TEXT}
<%= InputDefaultTypesHelper.getByInputDefaultTypesValue(attribute.getType().getValue()).getHelpText() %>
{AFTER_HELP_TEXT}
```

Figura 3.15: Trecho do *helper text*

O transformador faz uso do HTML5 e jQuery[21] para implementar funcionalidades acerca da usabilidade. Algumas *tags* HTML possuem atributos customizados iniciados por *data-metaerp* e o Javascript com o jQuery, através dos atributos, reconhece qual a verificação ou funcionalidade deve ser aplicada a esta tag. A Figura 3.16 mostra um trecho de código do transformador que indica que este campo deve ser obrigatório. Já a Figura 3.17 apresenta o trecho de código Javascript responsável por identificar os campos obrigatórios e tratá-los; este mesmo trecho verifica se algum campo possui uma expressão regular para validar o seu conteúdo.

```
if(attribute.isRequired()) {
    inputAttributes += " data-metaerp-required=\"true\"";
}
```

Figura 3.16: Trecho indicando a obrigatoriedade do campo

Os atributos *data* customizáveis do HTML5 são utilizados pelo transformador para implementar algumas funcionalidades de usabilidade de forma mais genérica e limpa (CONSORTIUM,). São utilizados inclusive a fim de satisfazer o FUF8, neste caso são utilizados cinco atributos customizáveis:

- *data-metaerp-entity*: a qual *entity* este campo está relacionado;
- *data-metaerp-attribute*: a qual *attribute* da *entity* este campo está relacionado;

[21] https://jquery.com/

```
var isRequired = $(el).attr('data-metaerp-required') !== undefined && $(el).attr('data-metaerp-required') === 'true';

if(isRequired && $(el).val() === '') {
    insertError(el, 'This field is required');
    return false;
} else if(regex) {
    insertError(el, 'The informed value is not valid');
    return false;
}
```

Figura 3.17: Trecho tratando a obrigatoriedade do campo

- *data-metaerp-main*: se este campo é o principal do seu grupo (*attributes related*);
- *data-metaerp-autocomplete*: se este campo será autocompletado quando o campo principal do grupo for informado;
- *data-metaerp-rule*: qual regra é aplicada a este campo.

Outros atributos customizáveis são usados para organização interna da lógica da funcionalidade. Estes atributos apresentados têm ligação direta com os campos do MetaUsaERPWeb e seu significado é exatamente o mesmo, apenas está sendo informado explicitamente a fim do Javascript gerar as funcionalidades.

É importante destacar que apesar de existir a verificação feita pela interface também existe algumas verificações servidoras garantindo que não houve alguma alteração no código *frontend* a fim de prejudicar as validações neste lado da aplicação. A Figura 3.18 apresenta o trecho do código do *controller* onde é feita a chamada ao método implementado no *model* correspondente (Figura 3.19) a fim de garantir que a *rule* definida é satisfeita. Os trechos entre as tags <% e %> e <%= e %> possuem relação com o modelo e ao longo da execução do transformador este é conferido e são realizadas as substituições necessárias.

A execução do *behavior*, de forma semelhante, é executado exclusivamente pelo *controller* e *model* correspondente. O *controller* executa algumas verificações e então realiza a chamada ao *model* que executa o *behavior*; as Figuras 3.20 e 3.21 apresentam trechos do código responsável por essa funcionalidade no *controller* e *model* respectivamente.

Alguns atributos são obrigatórios e outros possuem valores padrões definidos pelo transformador. No caso de atributos obrigatórios não informados em modelos sendo interpretados pelo transformador causam alerta de aviso ao usuário e a transformação é abortada.

3.3.4 Verificação do MetaUsaERPWeb

A fim de verificar a proposta (metamodelo e transformador) foi realizado um estudo de caso com escopo reduzido.

O escopo proposto é de uma empresa que revende produtos perecíveis e controlados. É de responsabilidade da empresa controlar o seu estoque e quais os vencimentos destes produtos. A empresa precisa apenas do controle da entrada de produtos conhecendo a quantidade total de produtos no estoque e lotes disponíveis; cada lote possui controle de quantidade, data de fabricação e data de validade.

```
if(modelRelated != "" && main != "") { %>
    $this->loadModel('<%= modelRelated %>');

    $keys = array_keys($this->request->data['<%= StringHelper.name2system(entityRelated.getName()) %>']);

    foreach($keys as $i) {
        $ruleCheck = $this-><%= modelRelated %>->checkRule($this->request->data
        ['<%= StringHelper.name2system(entityRelated.getName()) %>'][$i]['<%= main %>'],
        '<%= attributeName %>', '<%= rule %>', $this->request->data
        ['<%= StringHelper.name2system(entityRelated.getName()) %>'][$i]['<%= quantity %>']);

        if(!$ruleCheck) {

            $errorMessage = 'Rule check failed';

            if($internal) {
                return array(
                    'success' => false,
                    'error' => $errorMessage
                );
            }

            $this->Session->setFlash($errorMessage, 'flash_custom', array('class' => 'error'));
            $this->redirect(array('action' => 'create'));

        }

    }

<% }
```

Figura 3.18: Trecho no *controller* verificando a *rule*

```
function checkRule($id, $attribute, $rule, $value) {

    $data = $this->findById($id);

    if(!isset($data['<%= entityName %>'][$attribute])) {
        return false;
    }

    App::import('Vendor', 'NumberHelper');

    if($rule == 'bigger') {
        return $data['<%= entityName %>'][$attribute] < NumberHelper::numberTreatment($value);
    } else if($rule == 'biggerOrEqual') {
        return $data['<%= entityName %>'][$attribute] <= NumberHelper::numberTreatment($value);
    } else if($rule == 'equal') {
        return $data['<%= entityName %>'][$attribute] == NumberHelper::numberTreatment($value);
    } else if($rule == 'smaller') {
        return $data['<%= entityName %>'][$attribute] > NumberHelper::numberTreatment($value);
    } else if($rule == 'smallerOrEqual') {
        return $data['<%= entityName %>'][$attribute] >= NumberHelper::numberTreatment($value);
    } else if($rule == 'noRule') {
        return true;
    }

    return false;

}
```

Figura 3.19: Trecho no *model* verificando a *rule*

```
if(modelRelated != "" && main != "") { %>
    $this->loadModel('<%= modelRelated %>');

    $keys = array_keys($this->request->data['<%= StringHelper.name2system(entityRelated.getName()) %>']);

    foreach($keys as $i) {
        $behaviorCheck = $this-><%= modelRelated %>->executeBehavior($this->request->data
        ['<%= StringHelper.name2system(entityRelated.getName()) %>'][$i]['<%= main %>'],
        '<%= attributeName %>', '<%= behavior %>', $this->request->data
        ['<%= StringHelper.name2system(entityRelated.getName()) %>'][$i]['<%= quantity %>']);

        if(!$behaviorCheck) {

            $errorMessage = 'Relation update failed, please check your data: it may be wrong';

            if($internal) {
                return array(
                    'success' => false,
                    'error' => $errorMessage
                );
            }


            $this->Session->setFlash($errorMessage, 'flash_custom', array('class' => 'error'));
            $this->redirect(array('action' => 'create'));
        }

    }

<% }
```

Figura 3.20: Trecho no *controller* executando o *behavior*

```
function executeBehavior($id, $attribute, $behavior, $value) {

    $data = $this->findById($id);

    if(!isset($data['<%= entityName %>'][$attribute])) {
        return false;
    }

    App::import('Vendor', 'NumberHelper');

    if($behavior == 'add') {
        $data['<%= entityName %>'][$attribute] += NumberHelper::numberTreatment($value);
    } else if($behavior == 'subtract') {
        $data['<%= entityName %>'][$attribute] -= NumberHelper::numberTreatment($value);
    } else if($behavior == 'doNothing') {
        return true;
    }

    return $this->save($data);

}
```

Figura 3.21: Trecho no *model* executando o *behavior*

Assim este sistema ERP contará com dois módulos, produto e lote, integrados. O módulo de produto contará com as informações de cada medicamento e controlará o estoque com a ajuda do módulo de lote, portanto não é permitida a entrada de material através desse módulo. O módulo do lote contém as informações específicas do lote e se comunica com o módulo de

produto gerenciando o estoque.

São requisitos deste sistema:

R1. O sistema deverá gerenciar os produtos, contendo os seguintes campos em seu cadastro: nome, valor de venda e descrição;

R2. O sistema deverá gerenciar os lotes, contendo os seguintes campos em seu cadastro: identificação do lote, produto que o lote pertence, data de fabricação, data de validade e quantidade de produtos no lote;

R3. O sistema deverá, ao gerenciar a quantidade do lote atualizar a quantidade total de produto no estoque.

A Figura 3.22 representa o modelo conceitual dos elementos envolvidos no sistema e suas relações. Para a modelagem a partir do MetaUsaERPWeb utilizou-se o próprio *plugin* EMF que possui uma ferramenta para modelagem a partir do metamodelo criado.

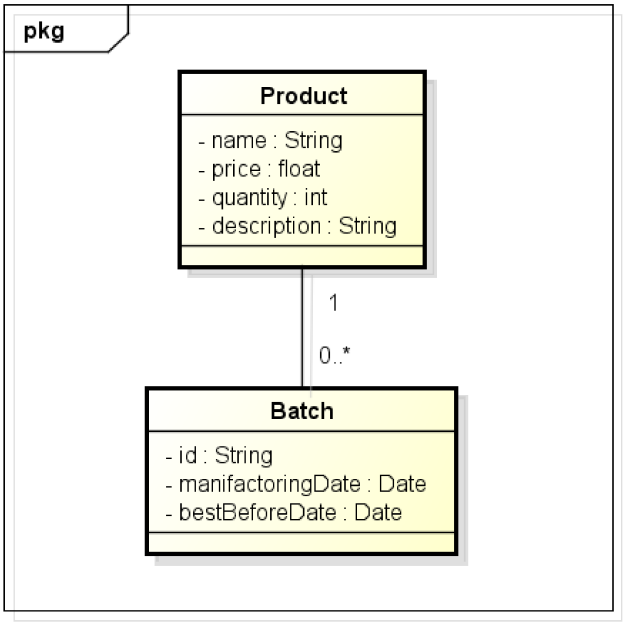

Figura 3.22: Diagrama de elementos esperados e suas relações

O modelo do escopo apresentado possui duas *entities*: *Product* e *Batch* que representam, respectivamente, o produto e o lote. A *entity Product* possui quatro *attributes*: *name*, *price*, *quantity* e *description*. A Tabela 3.7 demonstra como os atributos foram configurados no modelo; ao avaliá-la percebe-se que o *attribute quantity* não pode ser editável já que o controle de estoque é feito através dos lotes. Além disso o *description* não é editável, ou seja, não será exibido como coluna na listagem de produtos. Assim o requisito R1 está contemplado.

Já a *entity Batch* possui três atributos: *id*, *manifactoring date* e *best before*. A Tabela 3.8 apresenta a configuração dos atributos dos *attributes* desta *entity*. Percebe-se que o *attribute manufatoring date* não é obrigatório e não é apresentado em coluna na listagem de lotes. A *entity Batch* possui uma *entity related* que se relaciona a *entity Product*, denominada *BatchProduct*,

Tabela 3.7: Atributos dos *attributes* da *entity Product*

Atributo	*name*	*price*	*quantity*	*description*
Name	*Name*	*Price*	*Quantity*	*Description*
Label	*ProductName*	*ProductPrice*	*ProductQuantity*	*ProductDescription*
Type	*varchar*	*decimal*	*int*	*text*
Custom type	-	-	-	-
Creatable	*true*	*true*	*true*	*true*
Editable	*true*	*true*	*false*	*true*
Listable	*true*	*true*	*true*	*false*
Required	*true*	*true*	*true*	*false*

com o rótulo (*label*) *Product*; esta possui cardinalidade de 1. Ou seja, uma *entity BatchProduct* se relaciona a apenas uma *entity Product*. A *entity BatchProduct* possui dois *attributes related*: *product* e *quantity*; a Tabela 3.9 apresenta os seus atributos.

Tabela 3.8: Atributos dos *attributes* da *entity Batch*

Atributo	*id*	*manifacturing date*	*best before*
Name	*BatchId*	*BatchManufactoringDate*	*BatchBestBeforeDate*
Label	*Code*	*Manufacturing date*	*Best Before*
Type	*varchar*	*date*	*date*
Custom type	-	-	-
Creatable	*true*	*true*	*true*
Editable	*true*	*true*	*true*
Listable	*true*	*false*	*true*
Required	*true*	*false*	*true*

Tabela 3.9: Atributos dos *attributes* da *entity BatchProduct*

Atributo	*product*	*quantity*
Name	*BatchProductName*	*BatchProductQuantity*
Label	*Product*	*Quantity*
Type	*varchar*	*int*
Custom type	-	-
Creatable	*true*	*true*
Editable	*true*	*true*
Listable	*false*	*false*
Required	*true*	*true*
Related	*ProductName*	*ProductQuantity*
Autocomplete	*false*	*false*
Main	*true*	*false*
OnCreate	*doNothing*	*add*
OnDelete	*doNothing*	*subtract*
Rule	*noRule*	*noRule*

O *attribute BatchProductName* se relaciona ao *attribute ProductName* da *entity Product*; este *attribute* identifica qual produto será associado a este lote e, basiacamente, esta é a sua funcionalidade. Já o *attribute BatchProductQuantity* indica quantos produtos (especificados no *attribute* anterior) darão a entrada neste lote; o comportamento é definido no atribute *onCreate* e *onDelete* assim quando o lote for criado o valor definido neste *attribute* será adicionado ao *attribute ProductQuantity* (definido no atributo *related*), quando for deletado este valor será subtraído do *ProductQuantity*. O atributo *autocomplete* de ambos os *attributes* são *false*, logo não serão autocompletados quando o *attribute BranchProductName* (que está configurado como principal - *main* é *true*) for selecionado.

Em todos os *attributes* o atributo *custom type* não é informado já que é utilizado o atributo *type*.

Desta forma os campos estão de acordo com o requisito R2 e o comportamento está de acordo com o R3. A Figura 3.23 representa o esquema do modelo completo.

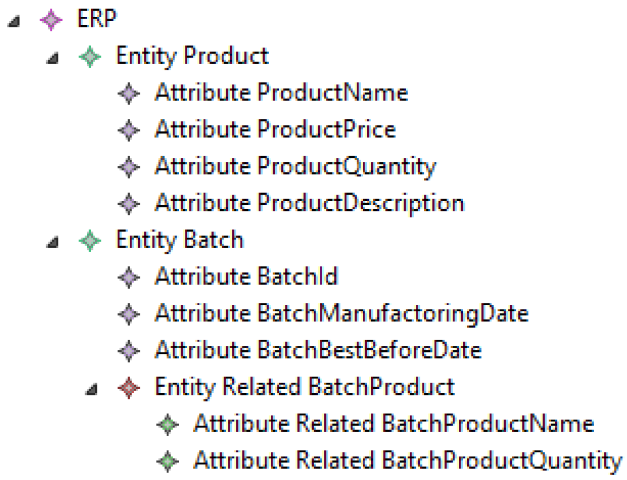

Figura 3.23: Esquema do modelo do estudo de caso completo

Após a conclusão do modelo foi realizado o processamento do transformador que gerou arquivos preparados para serem executados. Para este estudo de caso foi utilizado um *template* de interface qualquer apenas para fins visuais.

O transformador gerou três arquivos SQL para a criação das tabelas de banco de dados, um arquivo para cada tabela: *products*, *batches* e *batchproducts*. A Figura 3.24 apresenta o formulário gerado pelo transformador para a *entity Product*. A Figura 3.24(a) apresenta o formulário assim que a página é renderizada, é possível observar os campos, os rótulos e os textos de ajuda automaticamente incluídos pelo transformador a partir do modelo e da escolha correta dos tipos de campos. A Figura 3.24(b) representa a máscara automaticamente adicionada ao campo auxiliando o usuário no momento do preenchimento do formulário (FUF3), essa máscara é incluída graças ao *type* que o especifica. Por fim, a Figura 3.24(c) apresenta a validação do formulário que não permite que seja enviado sem o preenchimento dos campos obrigatórios.

Após enviar o formulário o usuário é redirecionado a listagem da *entity* (Figura 3.25) onde recebe uma mensagem de *feedback* de acordo com o retorno da ação de adicionar o registro ao banco de dados. A tabela de produtos já cadastrados apresenta as três colunas definidas como

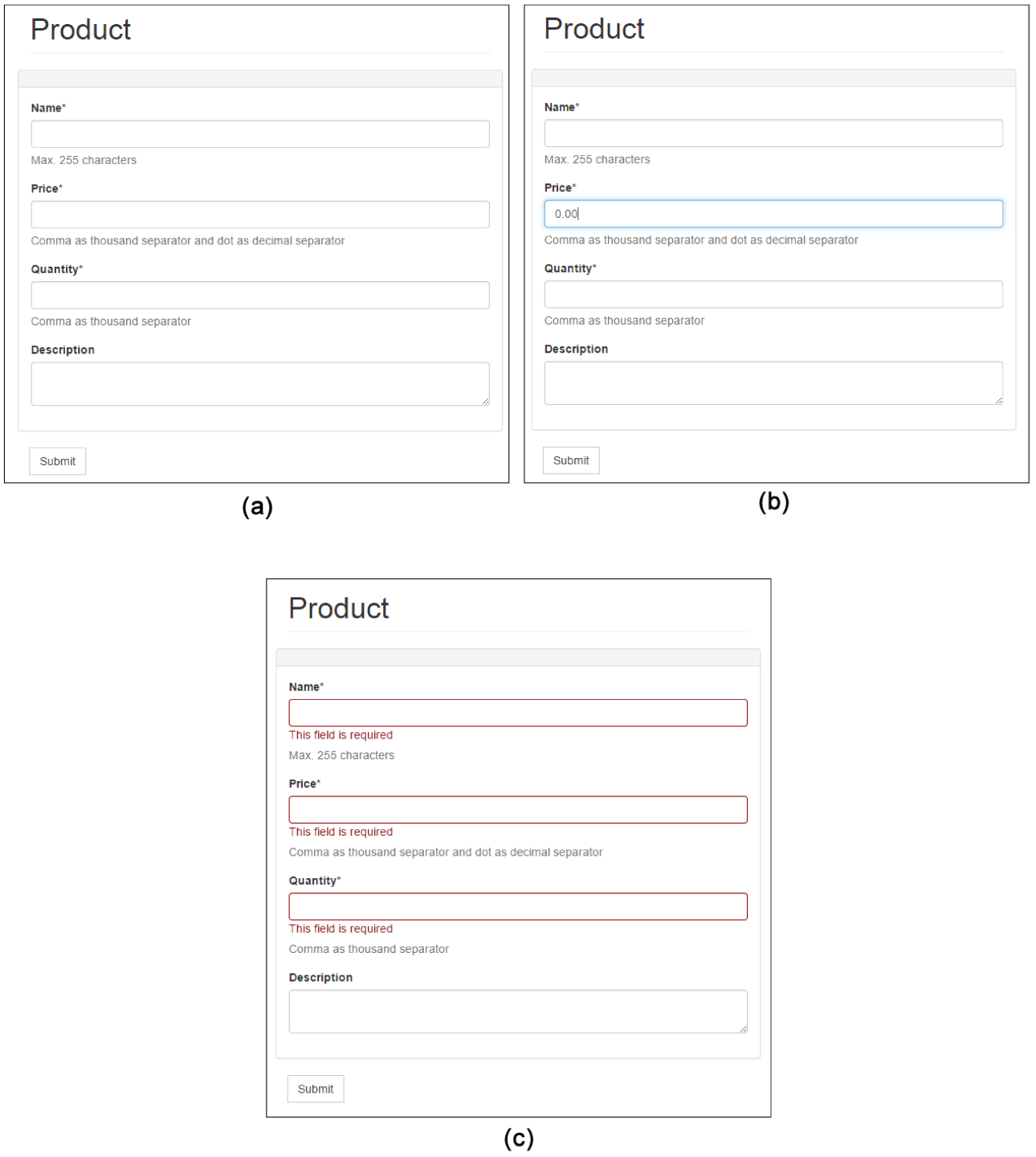

Figura 3.24: Formulário de adição da *entity Product*

true no atributo *listable* dos *attributes*; existe também uma quarta coluna com as opções de editar e deletar o registro.

O transformador automaticamente inclui verificação de exclusão de registros. Assim quando o usuário clica em *Delete* antes da efetivação da operação é feita uma consulta ao usuário (FUF1) conforme Figura 3.26.

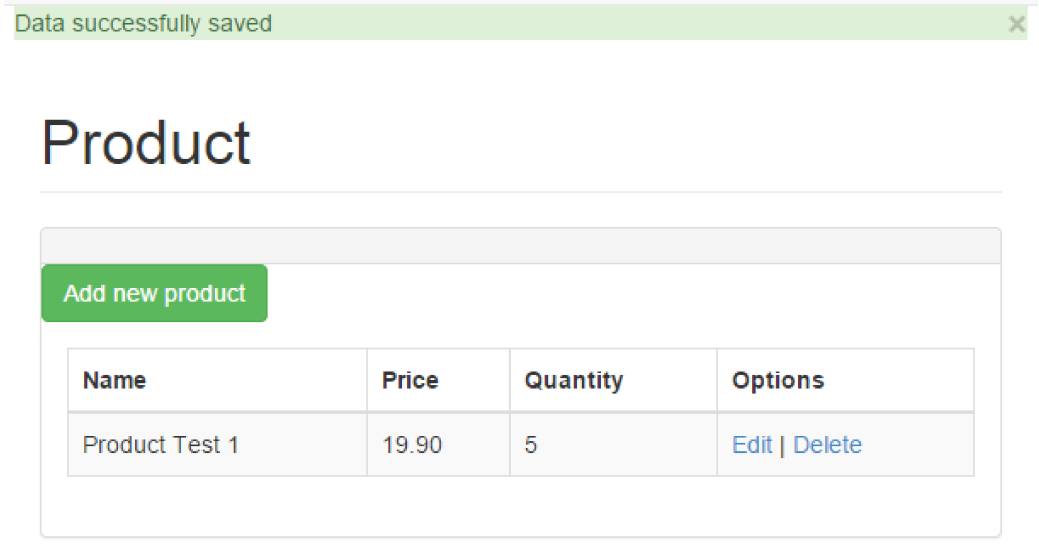

Figura 3.25: Listagem da *entity Product*

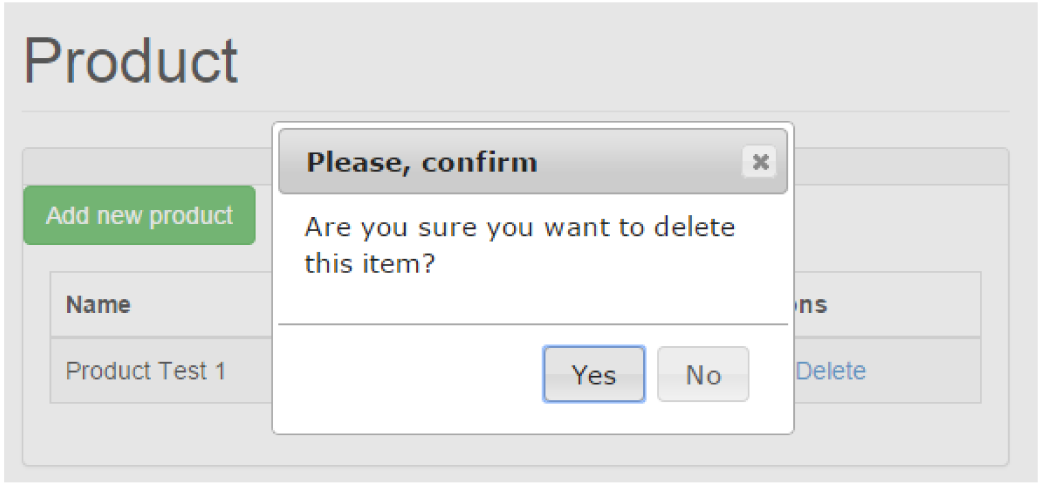

Figura 3.26: Confirmação da exclusão de registros

A Figura 3.27 apresenta o formulário de edição da *entity Product*. O formulário é muito parecido com o formulário de adição porém já está com os campos preenchidos com os valores atuais e não possui o *attribute Quantity* já que este teve o atributo *editable* configurado como *false*.

O formulário de cadastro da *entity Batch* é apresentada na Figura 3.28; assim como do formulário de adição da *entity Product*, possui os atributos definidos no modelo. Um detalhe que este formulário possui *attributes* do *type date*, nesse caso automaticamente o campo é formatado para receber a data e possui um calendário (Figura 3.28(b)); além disso também possui texto de ajuda para o formato correto a ser informado (FUF3).

O formulário possui um *box* extra que representa a *entity related BatchProduct*, é nele que são definidos os valores que estão conectados a *entity Product*. A Figura 3.29 demonstra como é seleção do produto no lote, basta o usuário digitar algumas letras do nome do produto para

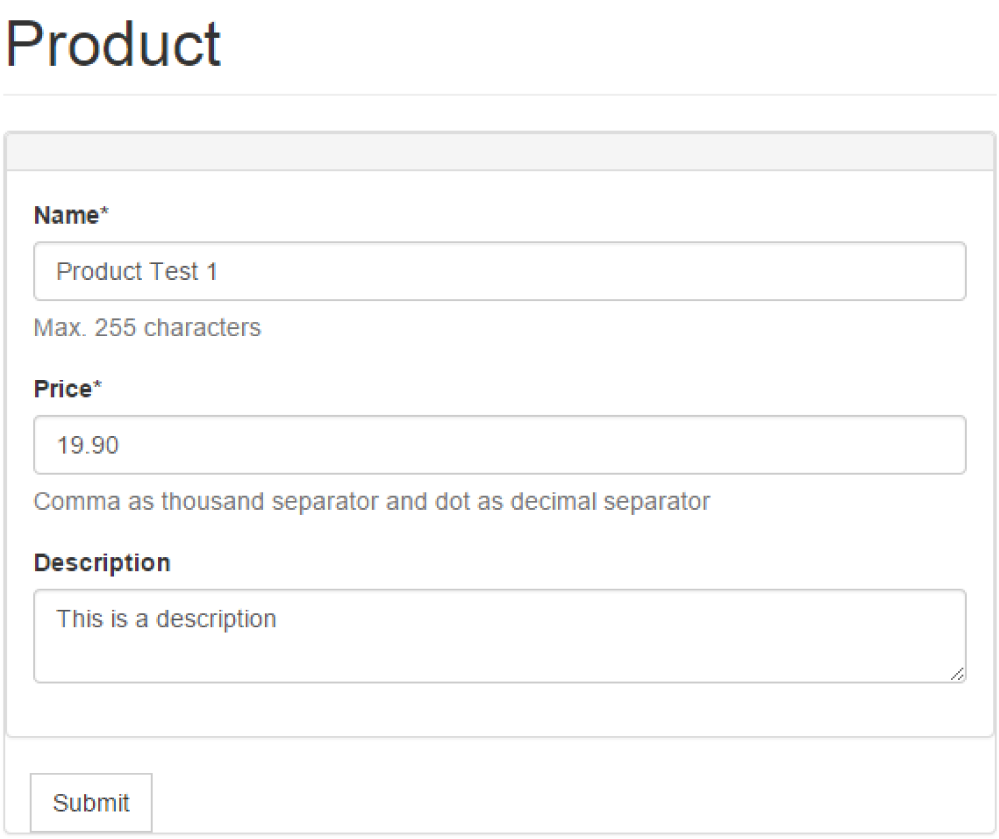

Figura 3.27: Formulário de edição da *entity Product*

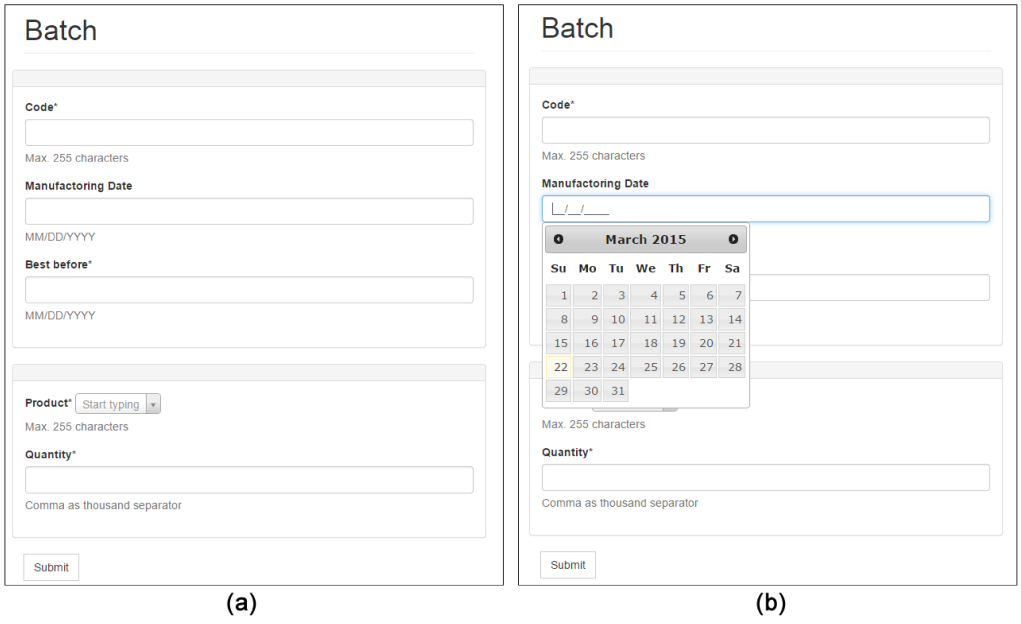

Figura 3.28: Formulário de adição da *entity Batch*

as opções serem exibidas; deste modo o usuário não precisa decorar o nome exato dos produtos nem qualquer outro código.

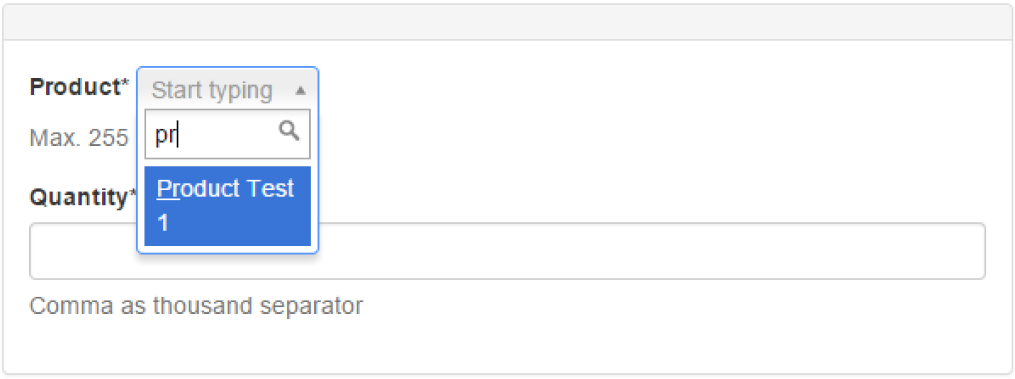

Figura 3.29: Detalhe para associar o produto ao lote

Ao finalizar essa operação a *entity Product* é automaticamente atualizado somando-se o valor do *attribute BatchProductQuantity* ao *attribute ProductQuantity*. Se este produto foi excluído então acontece a subtração deste valor.

3.4 Considerações finais

Este capítulo apresentou o MetaUsaERPWeb com objetivo de dar suporte ao desenvolvimento de um sistema ERP na Web no domínio de varejo considerando as características funcionais de usabilidade que são aderentes a estes sistemas. Foi apresentado o metamodelo MetaUsaERPWeb que permite a criação de modelos com lógicas de negócio específicas para esse tipo de software. Além do viés para regras de usabilidade, portanto o modelo possui definições deste tipo de característica.

Além da definição do metamodelo esse trabalho também criou um transformador M2C que recebe o modelo PIM resultante da modelagem baseada no MetaUsaERPWeb e gerando um código executável. Este código é baseado na plataforma Web, implementado na linguagem PHP utilizando a framework CakePHP na arquitetura MVC. Além disso também são gerados os códigos da interface gráfica utilizando-se a linguagem de marcação HTML, em sua versão 5, CSS e Javascript. O transformador permite a alteração de *layout* através de configurações, ou seja, não é necessária a alteração no código do transformador. Por fim também são gerados os *scripts* para criação e configuração do banco de dados utilizando-se o SGBD MySQL.

Capítulo 4

Validação da proposta

4.1 Considerações iniciais

A Engenharia de Software Experimental provê importantes métodos para testar hipóteses sobre uma nova tecnologia observando se os efeitos da sua adoção estão alinhados com o que foi declarado nas hipóteses (TRAVASSOS; GUROV; AMARAL, 2002).

Este trabalho baseou-se no estudo feito por Cirilo et al. (2011) que validou o estudo da metamodelagem para desenvolvimento de interfaces ricas. Além das análises realizadas por Cirilo et al. (2011) este trabalho apresenta outras que estão em consonância com seus objetivos.

Foram executados alguns tipos de análises nesta dissertação: análise de eficiência da proposta, análise de métricas, aceitação de tecnologia e avaliação heurística de usabilidade.

A próxima seção deste capítulo apresenta o estudo de caso aplicado para a validação deste trabalho; a seção seguinte apresenta a análise dos resultados após a aplicação do estudo de caso. Por fim a última seção aborda as considerações finais desta análise.

4.2 Estudo de caso

A hipótese principal de um experimento se chama hipótese nula e declara-se que não há um relacionamento estatisticamente significante entre a causa e o efeito que se quer investigar, ou seja, a única justificativa para o fenômeno observado seria apenas a casualidade ou coincidência. O objetivo principal do experimento é rejeitar a hipótese nula (H_0) em favor de uma ou mais hipóteses alternativas (H_1, H_2, H_3,...). A decisão de rejeição da hipótese nula deve ser balizada pela verificação dos resultados obtidos no experimento.

Existem dois tipos de variáveis: independentes e dependentes. As primeiras são manipuladas e controladas durante o experimento, as outras estão sob análise. Deve-se observar suas alterações com base nas mudanças feitas nas variáveis independentes. Essas variáveis que são manupuladas no experimento são chamadas fatores e apresentam a causa que afeta o resultado do processo de experimentação.

O estudo de caso consistiu no desenvolvimento de um ERP para a necessidade de uma empresa fictícia, baseado em um cenário, utilizando duas abordagens diferentes: uma usando o MetaUsaERPWeb e outra usando a implementação tradicional. Os trabalhos e os dados

coletados de ambas abordagens foram comparadas a fim de verificar a eficiência e eficácia da proposta desta dissertação.

Cada encontro do estudo de caso consistia em uma aula de três horas e quarenta minutos. No primeiro encontro foi realizado a preparação teorica, no encontro seguinte foi executado um estudo de caso de exemplo para melhor entendimento dos participantes. No último encontro o estudo de caso foi executado pelos participantes divididos em seus respectivos grupos, cada um com a sua metodologia previamente definida. As sub-seções seguintes detalham esses encontros e o estudo de caso em si.

O objetivo do experimento foi analisar o uso do MetaUsaERPWeb para o desenvolvimento de sistemas ERP de varejo empregando características funcionais de usabilidade, cujo propósito de avaliação era verificar a eficiência das equipes em termos de tempo despendido e produtividade do ponto de vista de desenvolvedores de software dentro do contexto acadêmico, estudantes de graduação e mestrado em Ciência da Computação.

4.2.1 Preparação e caracterização dos participantes

O estudo de caso foi executado com a turma da disciplina de Desenvolvimento para a Web no curso de gradução e mestrado em Ciência da Computação da Universidade Federal de São Carlos (UFSCar) - *campus* Sorocaba, durante três encontros, entre maio e junho de 2014. Fez parte do estudo de caso 4 alunos de mestrado e 50 alunos de graduação, a maioria entre o terceiro e quarto ano, divididos em 14 grupos de três ou quatro integrantes.

Antes da execução do estudo de caso todos os participantes preencheram um formulário com uma série de questões (Apêndice A), entre elas duas foram usadas para guiar a divisão dos grupos: "Qual é a sua experiência em linguagem de programação PHP?"e "Qual é a sua experiência com a técnica de Avaliação Heurística?".

A primeira questão possuia cinco respostas possíveis: nenhum; estudei em aula ou em livro; pratiquei em projetos em sala de aula; usei em projetos pessoais; na indústria ou uso em grande parte dos projetos que realizo.

A distribuição de respostas é apresentada na Figura 4.1, onde 33 participantes não possuem nenhum conhecimento em PHP, 9 participantes estudaram em aula ou livro, 7 participantes já usaram em projetos pessoais ou na indústria e 5 participantes já praticaram em projetos em sala de aula. Nenhum participante assinalou a última opção.

Já a segunda questão possuia quatro opções de resposta: eu não tenho familiaridade com este assunto, eu nunca fiz isto; li ou estudei sobre isto, conheço os conceitos e/ou técnicas; eu utilizo isto algumas vezes em projetos pessoais ou na indústria, mas não sou um(a) especialista; eu sou muito familiar com este assunto, eu me sentiria confortável fazendo isto.

A Figura 4.2 apresenta a distribuição de respostas: 41 participantes não possuiam familiaridade com o assunto, 9 leram ou estudaram sobre o assunto e 4 eram muito familiares com o assunto. A terceira opção não foi escolhida por nenhum participante.

O formulário foi composto por 27 questões, porém as outras 25 questões possuiam respostas muito próximas e não foram consideradas durante a divisão dos grupos. Nessa divisão existiu a preocupação em formar grupos ímpares com maior conhecimento em PHP e heurísticas de usabilidade, isso para garantir que os grupos da abordagem tradicional tivessem maior facilidade

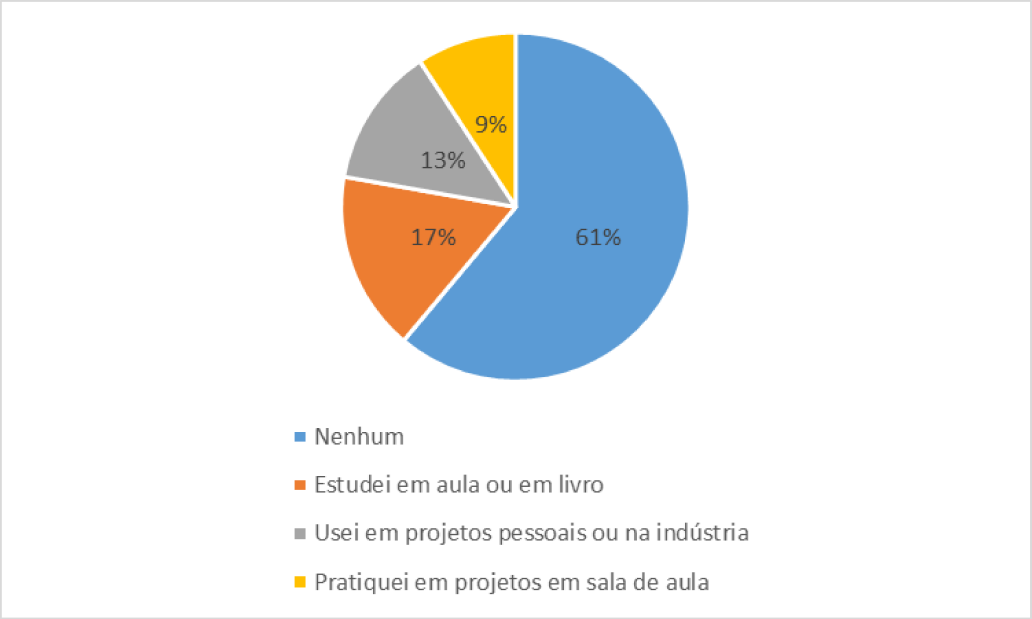

Figura 4.1: Distribuição de respostas sobre conhecimento em PHP

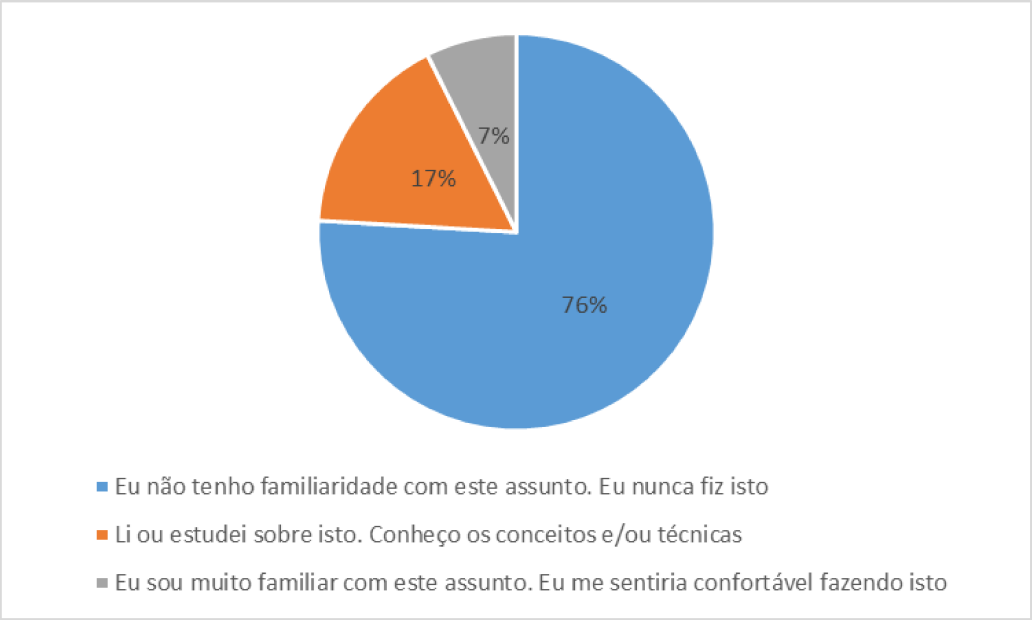

Figura 4.2: Distribuição de respostas sobre conhecimento em heurísticas

no uso das tecnologias já que sua abordagem era, em teoria, mais trabalhosa que dos grupos pares.

A fim de demonstrar de forma gráfica a divisão dos grupos atribuiu-se valores para cada opção respondida das questões. A Tabela 4.1 demonstra os valores atribuídos para cada opção da questão de experiência do PHP, já a Tabela 4.2 é referente à questão de heurísticas. Usando os valores dessa tabela os integrantes e grupos foram quantizados nas Figuras 4.3 e 4.4, respectivamente os grupos ímpares e grupos pares. Cada integrante é representado pelo número do seu grupo seguido por um número sequencial. Neles as barras azuis representam o conheci-

mento PHP do integrante, as barras laranjas representam o conhecimento em heurísticas; a área cinza demonstra a média do grupo no conhecimento PHP e a linha amarela mostra a média no conhecimento em heurísticas do grupo.

Tabela 4.1: Valores associados às respostas da questão de experiência com PHP

Opção	Valor
Nenhum	0
Estudei em aula ou em livro	1
Pratiquei em projetos em sala de aula	3
Usei em projetos pessoais ou na indústria	5

Tabela 4.2: Valores associados às respostas da questão de experiência com heurísticas

Opção	Valor
Eu não tenho familiaridade com este assunto. Eu nunca fiz isto	0
Li ou estudei sobre isto. Conheço os conceitos e/ou técnicas	1
Eu sou muito familiar com este assunto. Eu me sentiria confortável fazendo isto	5

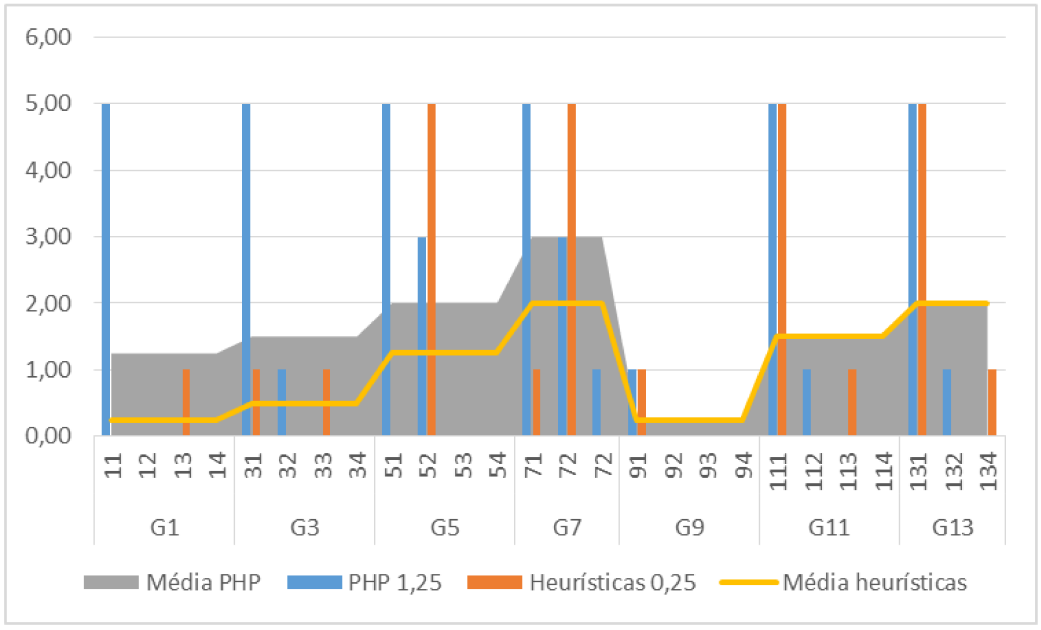

Figura 4.3: Distribuição dos grupos ímpares com base no conhecimento

Nesse gráficos pode-se perceber que os grupos ímpares e pares possuem diferença no nível de conhecimento, essa diferença foi proposital já que, em teoria, os grupos ímpares teriam maior esforço na codificação e na preocupação com as heurísticas. Já os grupos pares tinham foco apenas na modelagem da aplicação.

O primeiro encontro consistiu na preparação teórica dos participantes para a execução do estudo de caso. Foram debatidos os seguintes pontos: o objetivo do estudo de caso, o que é um ERP e alguns exemplos, o que são heurísticas de usabilidade e como elas se aplicam ao

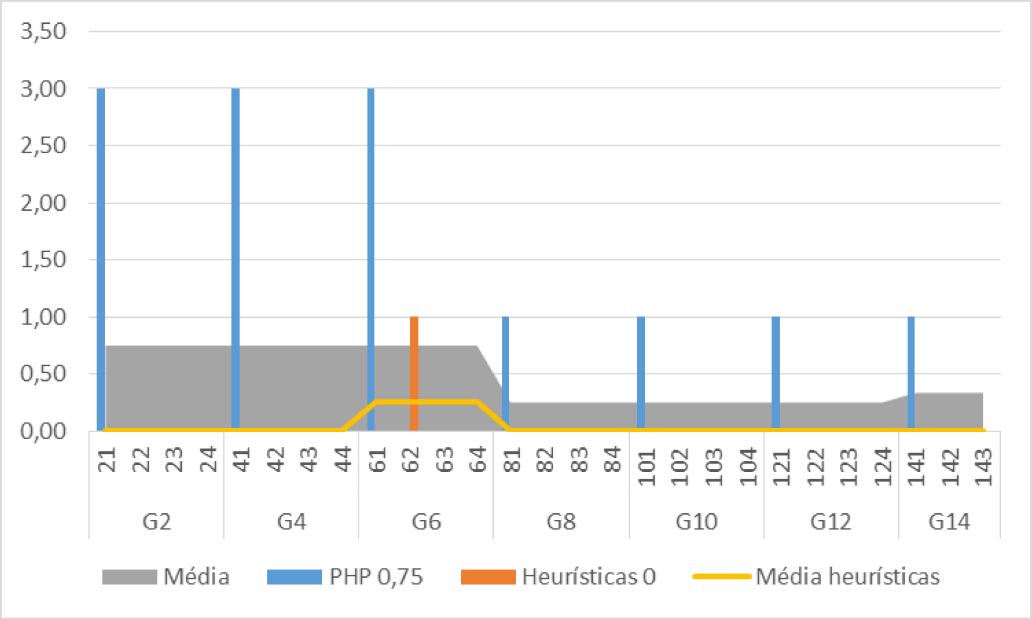

Figura 4.4: Distribuição dos grupos pares com base no conhecimento

assunto, como funciona o Desenvolvimento Dirigido a Modelos e o que é Modelagem Específica de Domínio. Os participantes já tinham conhecimento em alguns destes assuntos pois foram alvo de estudo durante a disciplina. Também foi explicado e exemplificado o uso das tecnologias Web aplicadas neste trabalho. Durante a disciplina os alunos já havian estudado sobre HTML, CSS e Javascript, portanto apenas uma revisão foi feita. Como na disciplina o foco é o desenvolvimento *server-side* na linguagem Java[1] houve uma preocupação maior na preparação dos participantes para entender melhor a linguagem usada no transformador proposto, o PHP.

O segundo encontro teve como objetivo aquecer os participantes para execução do estudo de caso. Esse aquecimento iniciou com a explicação teorica sobre o MetaUsaERPWeb, sendo apresentado o metamodelo e suas características. Após essa etapa foi apresentado um cenário fictício (Apêndice B) para guiar a parte prática do aquecimento. Nesse cenário uma empresa distribuidora de medicamentos estava em busca de um ERP para controlar os lotes de produtos recebidos. O sistema possuía como requisito dois módulos: de produto e lote, onde o segundo gerava a entrada no estoque e estava diretamente associado ao primeiro. Portanto, no fluxo de trabalho da empresa a área de recebimento informava a entrada de um lote no sistema e este calculava o novo valor de estoque com base no produto e quantidade do lote.

Os participantes foram acompanhados na modelagem e, ao final, o modelo (Figura 4.5) foi gerado. A partir desse modelo os participantes foram guiados na geração automática de código, criação do banco de dados e execução da aplicação.

Finalizando o segundo encontro foram explanadas os passos que deveriam ser executados no estudo de caso para envio correto do material gerado além de como preencher os formulários de acompanhamento. A execução do experimento foi finalizada no terceiro encontro conforme descrito a seguir.

[1]http://java.com/

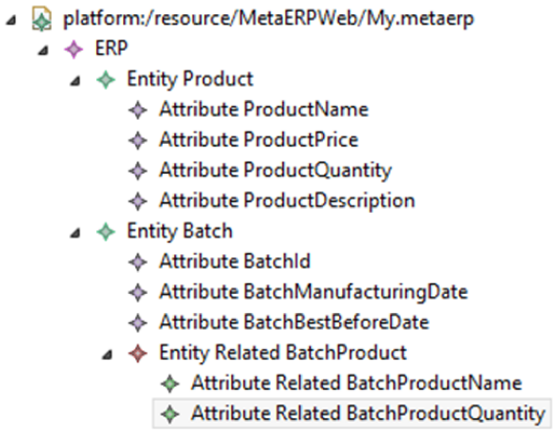

Figura 4.5: Modelo gerado no aquecimento

4.2.2 Execução do estudo de caso

O estudo de caso foi executado no terceiro e último dia de encontro. Os participantes, que assinaram o Termo de Consentimento Livre e Esclarecido (TCLE), separados em 14 grupos receberam as informações de como estavam divididos: os grupos ímpares deveriam executar o estudo de caso através da abordagem tradicional, ou seja, sem o suporte do metamodelo. O desenvolvimento deveria ser feito em PHP usando a *framework* CakePHP; já os grupos pares deveriam desenvolver a solução através do MetaUsaERPWeb.

O cenário apresentado (Apêndice C) foi de uma empresa fictícia de distribuição solicitando a implementação de um sistema ERP para controle de vendas e estoque. O sistema possuía como requisito principal que as vendas descontassem do estoque a quantidade vendida sem jamais permitir que o estoque se tornasse negativo, ou seja, deveria verificar se existia estoque suficiente para a quantidade vendida antes de confirmar a operação. Assim foram mapeados dois módulos: produto e venda, a Figura 4.6 apresenta a sugestão de modelo conceitual apresentada aos participantes para facilitar a compreensão do cenário proposto.

A complexidade do cenário foi limitada ao tempo de execução do estudo de caso, já que os participantes tinham o período de uma aula (3 horas e 40 minutos) para a implementação (tanto dos grupos pares como dos grupos ímpares) deveria ser possível nesse tempo.

Os grupos ímpares receberam alguns artefatos de apoio:

- Banco de dados: arquivo SQL com uma sugestão de estrutura de banco de dados da aplicação completa, a estrutura poderia ser alterada ou rejeitada pelos participantes se julgassem conveniente;

- Código base: estrutura do CakePHP com o módulo de produto completamente implementado e com o esqueleto do módulo de venda, essa estrutura era apenas uma sugestão e poderia ser alterada ou rejeitada pelos participantes;

- Template do sistema: o código base já continha um template padrão (*open-source*), mas também foi entregue aos participantes um arquivo com o template completo com outras implementações visuais;

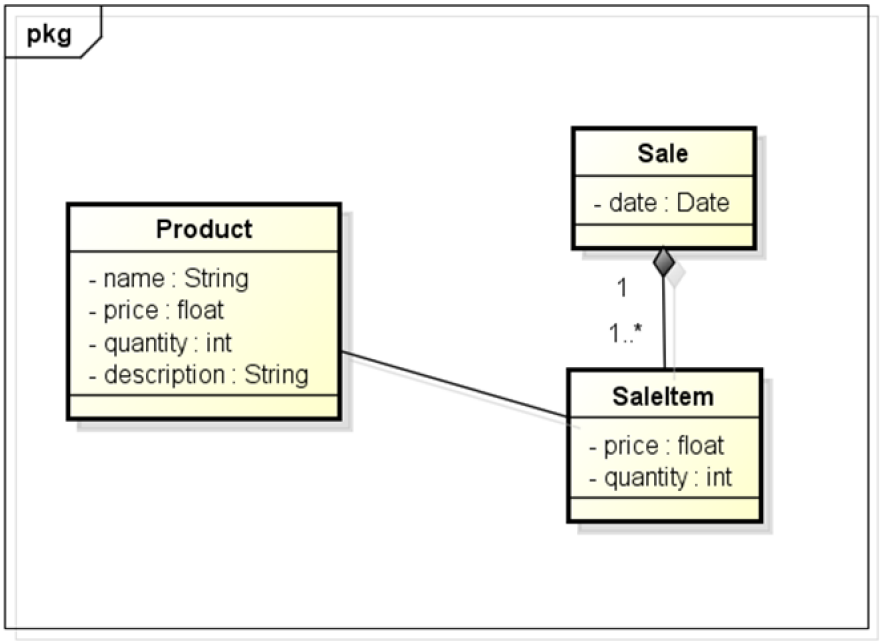

Figura 4.6: Diagrama de classes sugerido para execução do estudo de caso

- Material de apoio (Apêndice D): resumo sobre implementação com o PHP e CakePHP; explicação sobre o que consiste o código base e como deveria ser instalado; passos a seguir (planejamento, implementação e testes); forma de entrega dos materiais gerados.

Os grupos pares também receberam o material de apoio (Apêndice E) onde continha um resumo sobre como funcionava o MetaUsaERPWeb, seu metamodelo, entidades e relações; forma de execução dos arquivos de atualização do transformador, onde foram corrigidas algumas falhas; forma de entrega dos materiais gerados.

Todos os grupos receberam dois formulários de preenchimento obrigatório, formulário de coleta de dados (Apêndice F para os grupos tradicionais e Apêndice G para os grupos MetaUsaERPWeb) e formulário de avaliação (Apêndice H para os grupos tradicionais e Apêndice I para os grupos MetaUsaERPWeb). O primeiro deveria ser preenchido pelo grupo ao longo do estudo de caso com informações ligadas ao tempo levado em cada tarefa do desenvolvimento, número de linhas gerados automaticamente e manualmente, além de reportar problemas encontrados durante a implementação. O segundo formulário, preenchido individualmente ao final da atividade, questionava os participantes sobre qual foi a experiência deles durante o estudo de caso além de um espaço para sugestões e observações. Os formulários foram adaptados para a realidade da abordagem de cada grupo, porém a essência das questões era a mesma.

A seguir apresenta-se as tarefas executadas pelos grupos ao longo do desenvolvimento. Em ambas as abordagens os grupos deveriam executar três tarefas de desenvolvimento:

Projetar: Na tarefa Projetar os participantes estudaram e discutiram o cenário e a estrutura da solução. No caso dos grupos ímpares essa tarefa contemplou a análise do código base e sua adesão à solução imaginada pelo grupo. Estes grupos também poderiam gerar modelos que representassem a sua solução porém essa etapa não era obrigatória. Já os grupos pares deveriam,

nessa fase, gerar o modelo a partir do MetaUsaERPWeb que representasse uma solução para o cenário proposto.

Implementação: Nessa tarefa os participantes deveriam codificar a solução. Para os grupos pares essa etapa consistia na execução do transformador de código, instalação do banco de dados e alguma outra implementação adicional que julgassem pertinente não gerada automaticamente pelo transformador. Já os grupos ímpares deveriam fazer a implementação manual da solução podendo ter a ajuda de bibliotecas abertas.

Testes: Nessa tarefa os participantes testaram a solução. Os erros encontrados pelo grupos ímpares deveriam ser corrigidos. Já os grupos pares deveriam reportar os erros que foram originados do transformador para que fossem corrigidos posteriormente na origem, enquanto isso os erros provenientes de implementação manual também deveriam ser corrigidos. A execução dos testes poderia ser feito em qualquer navegador de qualquer sistema operacional já que seu funcionamento deve ser universal.

Ao finalizar o estudo de caso os materiais gerados foram recolhidos e analisados de formas diferentes a fim de validar o processo sob diferentes focos.

4.3 Análise dos resultados

Cada análise executada possui uma questão de pesquisa e, em alguns casos, possíveis hipóteses onde apenas uma hipótese é selecionada balizada nas métricas e análises apresentadas nas próximas sub-seções.

As variáveis independentes selecionadas para o experimento foram o processo de desenvolvimento, aplicação desenvolvida, ambiente de desenvolvimento e tecnologias de desenvolvimento. E a variável dependente foi a questão de pesquisa de cada análise. O fator estabelecido foi o processo de desenvolvimento, as outras variáveis independentes se manteram constantes.

4.3.1 Análise de eficiência da proposta

Para a análise fez-se a seguinte questão de pesquisa: *há diferença na eficiência das equipes MetaUsaERPWeb e tradicional?* A seguintes hipóteses são levantadas:

- Hipótese nula (HExp1_0): Não há diferença significativa (maior ou igual a 10%) na eficiência das equipes MetaUsaERPWeb e tradicional;
- Hipótese alternativa (HExp1_1): As equipes MetaUsaERPWeb são significativamente (maior ou igual a 10%) mais eficientes do que as equipes tradicionais;
- Hipótese alternativa (HExp1_2): As equipes tradicionais são significativamente (maior ou igual a 10%) mais eficientes do que as equipes MetaUsaERPWeb.

Utilizou-se as seguintes métricas para a análise:

- τ: tempo total, em minutos, gasto pela equipe para o desenvolvimento dos módulos;

- ϕ: produtividade da equipe em termos de linhas de código produzidas (LOC - *Lines Of Code*) por unidade de tempo, ou seja, ϕ= LOC / τ;
- μ_{T}: tempo médio, em minutos, consumido pelas equipes para o desenvolvimento dos módulos;
- μ_{P}: produtividade média das equipes no desenvolvimento dos módulos.

Portanto pode-se dizer que a equipe A é mais eficiente que B se $\mu_{\mathrm{TA}} < \mu_{\mathrm{TB}}$ e $\mu_{\mathrm{PA}} > \mu_{\mathrm{PB}}$. De forma semelhante podemos dizer que não há diferença entre as equipes A e B se $\mu_{\mathrm{TA}} = \mu_{\mathrm{TB}}$ e $\mu_{\mathrm{PA}} = \mu_{\mathrm{PB}}$.

Tabela 4.3: Tempos consumidos pelos grupos

Abordagem	Grupo	HIproj	HTproj	HIimpl	HTimpl	HItes	HTtes	Total
Tradicional	G1	8:21	8:33	8:34	11:25	-	-	183
	G3	8:30	9:00	9:00	11:31	11:31	11:35	185
	G5	8:21	8:50	8:50	11:00	-	-	219
	G7	8:30	8:40	8:40	11:35	-	-	185
	G9	8:15	9:10	9:10	11:15	11:15	11:25	190
	G11	8:53	9:00	9:00	11:12	-	-	139
	G13	8:30	9:30	9:30	11:40	-	-	190
	Média	29		144		4		184,4
MetaUsaERPWeb	G2	8:25	9:00	9:00	9:30	9:30	10:00	95
	G4	8:33	8:56	8:56	9:30	9:30	10:20	107
	G6	8:30	9:30	9:50	10:10	10:10	11:10	140
	G8	8:30	9:05	9:05	9:25	9:25	9:50	80
	G10	8:38	9:17	9:18	9:50	9:51	10:43	123
	G12	8:24	8:54	8:54	9:36	9:36	10:40	136
	G14	8:25	8:40	8:40	9:25	9:25	9:50	85
	Média	33		31		43		109,4

Tabela 4.4: Legenda da Tabela 4.3

Legenda	Significado
HIproj	Horário de início da tarefa de projeto
HTproj	Horário de término da tarefa de projeto
HIimpl	Horário de início da tarefa de implementação
HTimpl	Horário de término da tarefa de implementação
HItes	Horário de início da tarefa de teste
HTtes	Horário de término da tarefa de teste
Total	Tempo total das três tarefas, em minutos

A Tabela 4.3 (legenda na Tabela 4.4) apresenta os dados coletados pelos grupos na execução do estudo de caso. Os grupos G1, G5, G7, G11 e G13 executaram a tarefa de testes juntamente com a implementação dos módulos, portanto omitiram o tempo gasto especificamente nela. A

média da tarefa de testes desconsidera esses grupos calculando apenas para os grupos G3 e G9. A Figura 4.7 representa o tempo médio gasto por tarefa por abordagem.

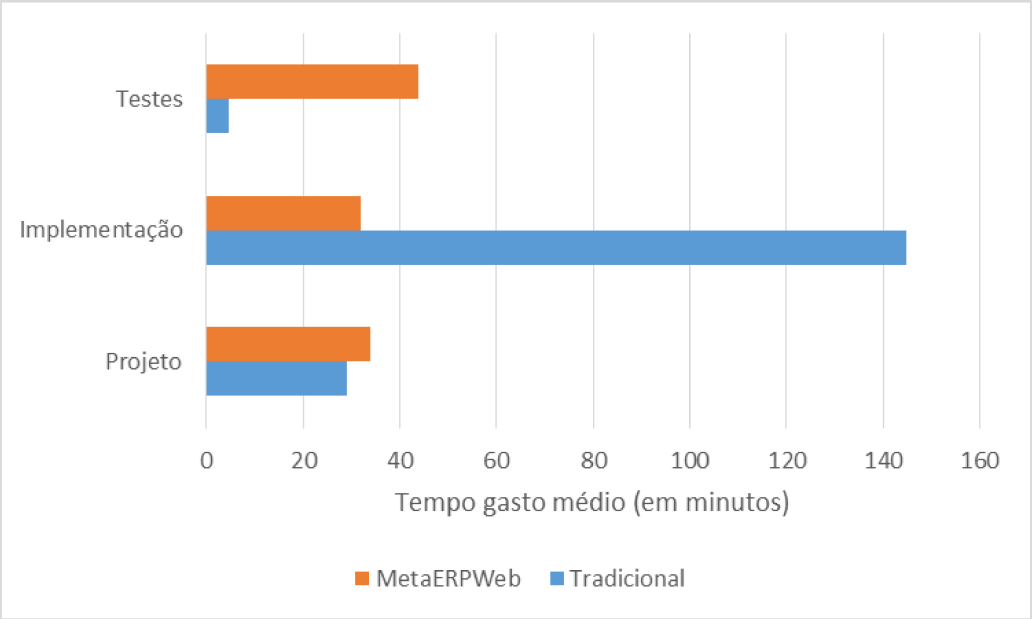

Figura 4.7: Tempo médio gasto por tarefa por abordagem

Nessa primeira apresentação já é possível observar que os grupos ímpares demoraram, em média, 184 minutos para completar a aplicação, enquanto os grupos pares consumiram 109 minutos para a mesma tarefa; aproximadamente 41% mais rápido. Também é interessante observar que os tempos de projeto e testes é maior para os grupos pares porém esse tempo maior é justificado na tarefa de implementação que é quase cinco vezes menor que o tempo gasto pelos grupos ímpares. No restante desse trabalho será utilizado apenas o tempo total para a análise.

Para análise da produtividade dos grupos a Tabela 4.5 apresenta os dados de linhas de código geradas em cada grupo juntamente com o tempo, em horas, que os grupos levaram para finalizar a aplicação bem como suas produtividades.

Apesar dos integrantes dos grupos terem informado o número de linhas produzido foi feita uma nova contagem, todas utilizando a mesma abordagem, para garantir a validade dos dados. Nessa contagem foram considerados apenas os códigos gerados nos *controllers*, *models* e *views* além do código SQL de geração da estrutura do banco de dados e o javascript gerados pelo transformador ou pelos integrantes. Os código gerados foram submetidos por uma formatação automática a fim de indentar o código eliminando linhas em branco desnecessárias que pudessem ter impacto no número total de linhas.

Os grupos ímpares, em média, produziram 2.026 linhas de código em 2 horas e 56 minutos levando a uma produtividade de 701 linhas de código por hora; enquanto isso cerca de 2.583 linhas de código foram produzidas em 1 hora e 49 minutos, em média, pelos grupos pares em uma produtividade de 1.478 linhas de código por hora. Portanto os grupos pares foram 111% mais eficientes que os grupos ímpares, além de produzir mais linhas de código esses grupos também foram mais rápidos para produzí-las.

Tabela 4.5: Produtividade de cada grupo

Abordagem	Grupo	LOC	Tempo (h)	Produtividade (ϕ)
Tradicional	G1	1.945	3,05	637,70
	G3	2.026	3,08	657,08
	G5	2.214	2,65	835,47
	G7	1.925	3,08	624,32
	G9	1.960	3,17	618,95
	G11	2.068	2,32	892,66
	G13	2.040	3,17	644,21
	Média	2.025,43	2,93	701,49
MetaUsaERPWeb	G2	2.662	1,58	1.681,26
	G4	2.643	1,78	1.482,06
	G6	2.640	2,33	1.131,43
	G8	2.529	1,33	1.896,75
	G10	2.524	2,05	1.231,22
	G12	2.511	2,27	1.107,79
	G14	2.571	1,42	1.814,82
	Média	2.582,86	1,82	1.477,90

Outra forma de calcular a produtividade dos grupos pares é calcular a razão entre a especificação e o código (REC), essa métrica permite determinar a relação entre um elemento de especificação e o código gerado correspondente (LUCRÉDIO, 2009). Por exemplo, se a especificação de uma entidade de 10 linhas produzir 1.000 linhas de código tem-se relação 1 para 100; em outras palavras, cada linha de especificação gera 100 linhas de código. Dessa maneira é possível verificar o ganho de produtividade em termos de número de linhas, essa métrica é calculada da seguinte forma (onde NEE é número de elementos especificados):

$$REC = \frac{\sum LOC(cdigogerado)}{\sum NEE(modelos)} \tag{4.1}$$

A Tabela 4.6 apresenta os resultados dessa notação nesse estudo de caso. O grupo G10 não enviou o modelo usado para execução pelo transformador portanto foi retirado dessa análise. Percebe-se que todos os grupos usaram o mesmo número de linhas para definição do modelo, 17 linhas, porém as linhas geradas (LOC) foram diferentes para os grupos. Isso aconteceu pois definições de atributos, comportamentos e regras podem ser diferentes em uma única linha.

Os grupos obtiveram resultados muito próximos, sendo o grupo G2 mais produtivo com REC = 156,59 e o grupo G12 o menos produtivo com REC = 147,71 porém a diferença entre os dois é de apenas 8,88. Assim a média REC é de 152,51, ou seja, a cada linha definida no modelo gera aproximadamente 153 linhas automaticamente pelo transformador proposto. Assim conclui-se que o mesmo elemento especificado possui diferentes características que impactam na funcionalidade e, consequentemente, código gerado.

Para verificar de forma gráfica os tempos e as produtividades dos grupos além de possíveis *outliers* as Figuras 4.8 e 4.9 apresentam gráficos de caixa (*boxplot*) mostrando, respectivamente, a dispersão dos tempos totais e das produtividades. No *boxplot*, a linha sobre cada caixa marca

Tabela 4.6: Produtividade REC dos grupos pares

Grupo	LOC	NEE	REC
G2	2.662	17	156,59
G4	2.643	17	155,47
G6	2.640	17	155,29
G8	2.529	17	148,76
G12	2.511	17	147,71
G14	2.571	17	151,24
Média	2.582,86	17	152,51

a mediana do conjunto de dados e representa o segundo quartil (Q2) da distribuição. As partes inferior e superior são delimitadas pelos quartis inferior (Q1) e superior (Q3) respectivamente. As hastes que se estendem acima e abaixo das caixas representam o limite teórico dentro do qual provavelmente são encontrados todos os dados da amostra se a distribuição for normal. A haste inferior se estende do quartil inferior até o menor valor não inferior a Q1 - 1,5*IRQ, onde IRQ é o intervalo interquartil (Q3 - Q1). Já a haste superior se estende do quartil superior até o maior valor não superior a Q3 + 1,5*IRQ. Os valores fora desses limites, representados como pontos individuais no gráfico, são considerados *outliers*.

Observando-se as Figuras 4.8 e 4.9 é possível perceber que nenhum *oulier* foi idenficado. Também é possível observar claramente que a tendência dos grupos pares é realizar a implementação de forma mais rápida (Figura 4.8) e com maior produtividade (Figura 4.9), já os grupos ímpares tenderam a ser mais lentos e menos produtivos.

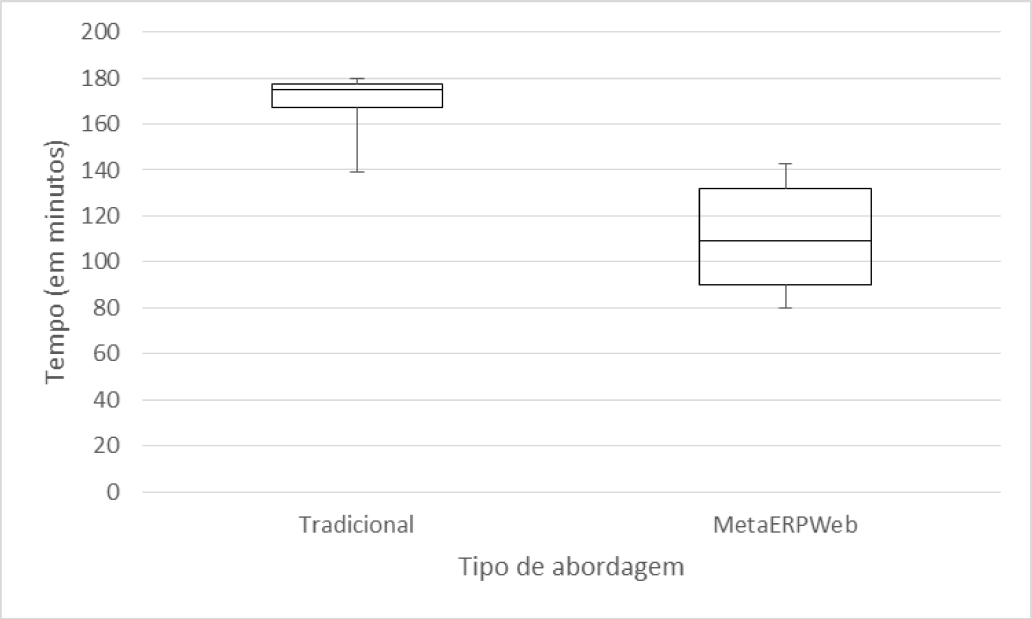

Figura 4.8: Boxplot de dispersão dos tempos totais

Utilizou-se o teste de normalidade de Shapiro-Wilk (MONTGOMERY, 2006) para verificar se a amostra obtida através do estudo de caso possui uma distribuição normal ou não. Deseja-se

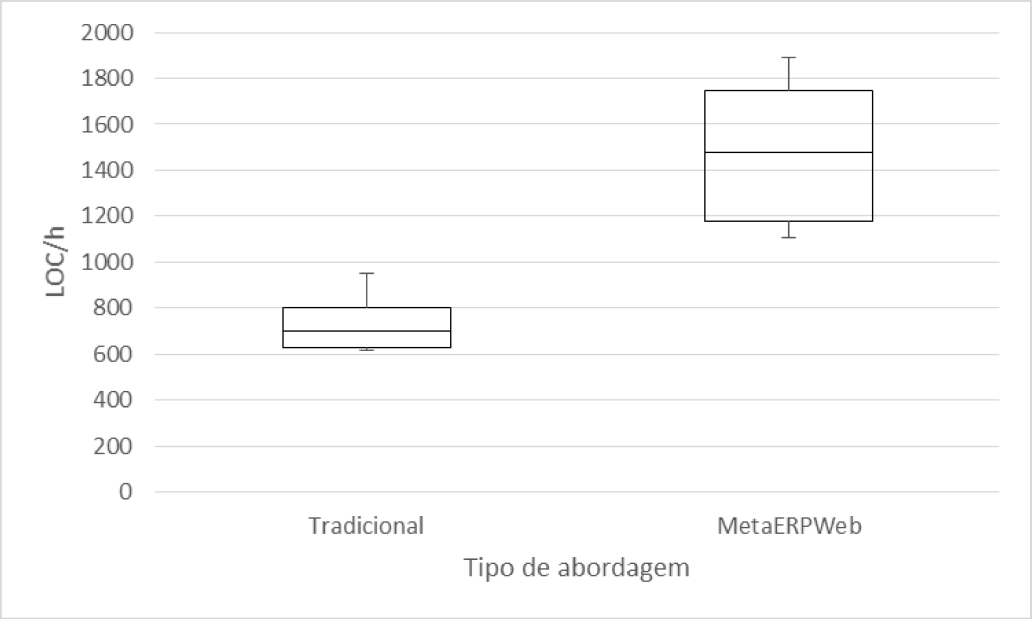

Figura 4.9: Boxplot de dispersão das produtividades

afirmamar com um grau de significancia de 5% que as amostras possuem distribuição normal.

A Figura 4.10 (a) apresenta os tempos totais do MetaUsaERPWeb e a reta normal da amostra, onde é possível verificar que os pontos estão próximos da reta em uma distribuição normal. Além disso a Estatística Shapiro-Wilk = 0,921 e o P-valor = 0,478; assim afirma-se com um grau de significância de 5% que a amostra de tempos totais dos grupos utilizando o MetaUsaERPWeb possui distribuição normal.

Já a amostra dos tempos totais obtida a partir dos grupos utilizando a abordagem tradicional apresentou a Estatística Shapiro-Wilk = 0,760 e P-valor = 0,016; assim não é possível afirmar que esta amostra possui distribuição normal. Este fato é reforçado pela Figura 4.10 (b) onde é possível observar a aleatoriedade dos pontos ao redor da reta normal.

O teste de normalidade de Shapiro-Wilk também foi executado nas amostras referentes a produtividade; a Figura 4.11 apresenta os pontos em relação a reta normal. A amostra obtida a partir dos grupos que utilizaram o MetaUsaERPWeb (Figura 4.11 (a)) possui Estatística Shapiro-Wilk = 0,857 e P-valor = 0,142; já a amostra adquirida a partir dos grupos utilizando a metodologia tradicional (Figura 4.11 (b)) possui Estatística Shapiro-Wilk = 0,895 e P-valor = 0,301. Portanto é possível afirmar, com 5% de significância, que ambas as amostras possuem distribuição normal.

Deseja-se verificar com algum grau de significância se é possível rejeitar a hipótese nula em favor de alguma das hipóteses alternativas com o conjunto de dados obtidos. Para comprovar o efeito verificado anteriormente de forma estatística aplicou-se o Teste-U de Mann-Whitney e o Teste-T (MONTGOMERY, 2006). Esses testes são usados para comparar duas amostras independentes e verificar se as médias de seus dados são estatisticamente diferentes e, portanto, provar que o efeito hipotético foi demonstrado. No caso desse estudo de caso as amostras foram constituídas pelos dados referentes aos tempos totais e produtividades dos grupos ímpares e dos grupos pares.

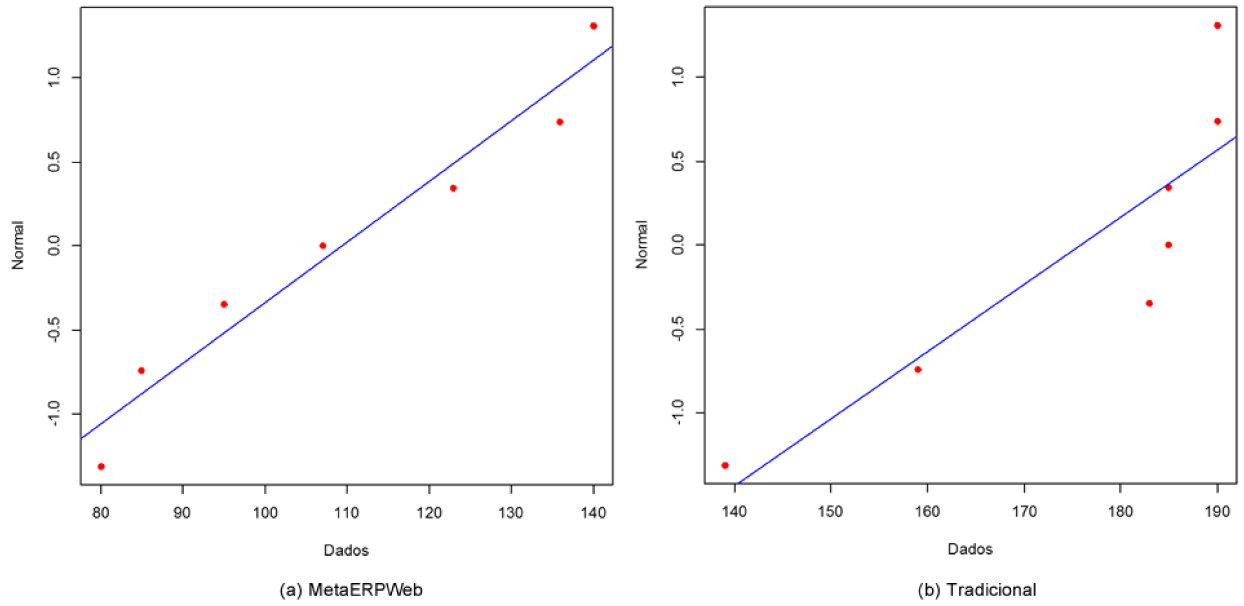

Figura 4.10: Teste de normalidade dos tempos totais

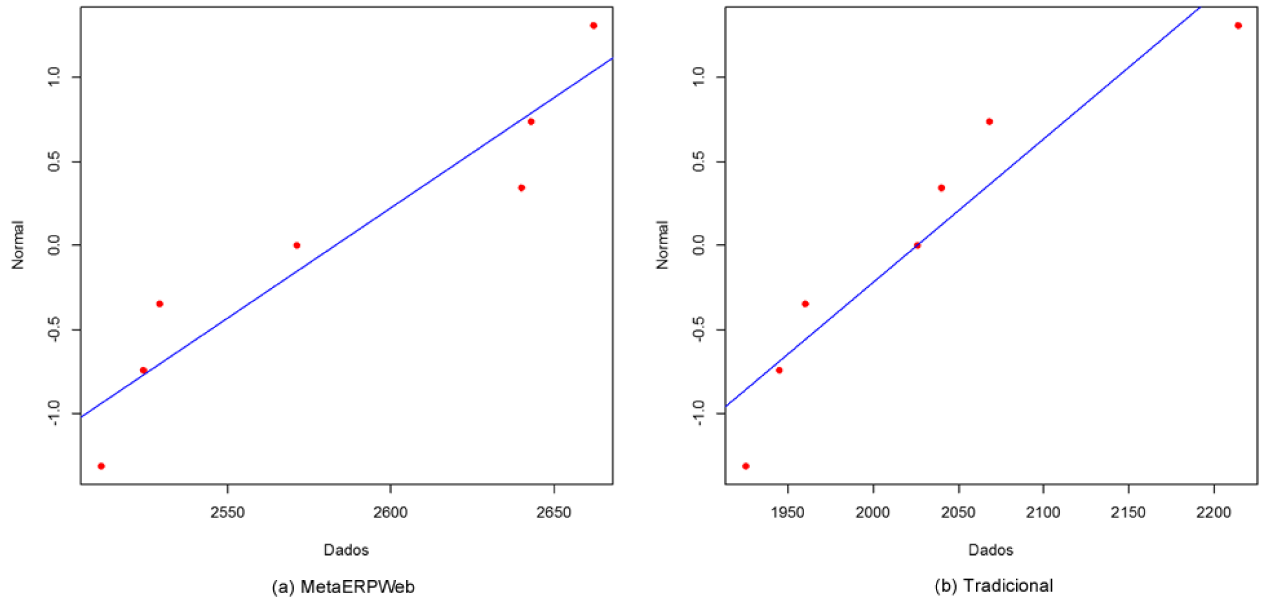

Figura 4.11: Teste de normalidade das produtividades

Conforme indicado pelas Figuras 4.10 e 4.11 nem todas as amostras possuem distribuição normal e o Teste-T requer, usualmente, esse tipo de distribuição. Apesar do Teste-T ser robusto o suficiente para suportar alguns desvios nesse requisito (WOHLIN, 2000; HART, 2001). Esta dissertação utilizou o Teste U de Mann-Whitney e, para verificar o resultado, também utilizou o Teste-T.

O Teste U de Mann-Whitney consiste em tratar as duas amostras dos grupos como se fosse uma só, a partir dessa junção os valores são ordenados de forma crescente e classificados em

postos. Ao final desta etapa considera-se S_m e S_n a soma dos postos relacionados aos elementos das amostras iniciais. A partir desta soma é possível obter os valores de U_m e U_n conforme as Equações 4.2 e 4.3 apresentam.

$$U_m = S_m - \frac{m(m+1)}{2} \tag{4.2}$$

$$U_n = S_n - \frac{n(n+1)}{2} \tag{4.3}$$

Por fim W é escolhido a partir de U_m e U_n dependendo do objetivo da análise (se a hipótese alternativa é referente a diferença das amostras, ou se uma amostra deve ser maior ou menor que a outra), a partir desse valor é possível obter o P-valor através das tabelas estatísticas do Teste U de Mann-Whitney.

O Teste-T baseia-se na ideia de que, quando se procura por diferenças entre duas amostras X e Y, deve-se julgar a diferença entre as suas médias considerando a dispersão ou variabilidade dos dados das amostras comparadas, maiores chances desses dados estarem sobrepostos na distribuição normal, mesmo que a diferença das médias se mantenha constante. A Equação 4.4 apresenta a fórmula do Teste-T. Trata-se da razão entre a diferença das médias e a medida da variabilidade ou dispersão dos dados.

$$t_0 = \frac{\overline{x} - \overline{y}}{S_P \sqrt{\frac{1}{n} + \frac{1}{m}}} \tag{4.4}$$

$$S_P = \sqrt{\frac{(n-1){S_x}^2 + (m-1){S_y}^2}{n+m-2}} \tag{4.5}$$

Após o cálculo de t_0 deve-se verificar a tabela padrão de distribuição de probabilistica estatística de t de Student (GOSSET et al., 1943) a fim de verificar se a razão calculada é suficientemente grande de modo que se possa atestar que é improvável que a diferença amostral tenha sido mera casualidade. Estabelece-se um grau de significância αque represente o "nível de risco"ou "ponto de corte"com o qual é permitido afirmar que há diferença entre as amostras e rejeitar a hipótese nula. Por exemplo, tomando-se α= 0,05 significa que se assume um risco de 5% em se encontrar uma diferença significativa entre as médias das amostras, mesmo que essa diferença fosse falso positivo, ou seja, por acaso. Então o nível de confiança do resultado do teste neste caso seria de 95%.

Já que o efeito na variável dependente do experimento (eficiência da equipe) envolveu dois aspectos, tempo total (τ) e produtividade (ϕ), a aplicação dos testes ao conjunto amostral de dados foi realizada em duas etapas. Na primeira etapa compararou-se as amostras relativas aos tempos totais de cada grupo; na segunda etapa foram comparadas as amostras referentes a produtividade do grupo. O teste da hipótese nula baseou-se na combinação dos critérios de rejeição de ambas etapas do teste de modo que H_0 apenas seria rejeitada se, e somente se, pudesse ser rejeitada nas duas etapas do teste. Para os propósitos dessa análise em cada etapa do teste utilizou-se o menor grau de significância αcom o qual fosse possível rejeitar a hipótese nula, assumindo-se um grau de significância máximo de 5%.

A primeira etapa consiste na análise do tempo total do experimento, tem-se como entrada duas amostras independentes: $X_{\tau \text{MetaUsaERPWeb}}$= {95, 107, 140, 80, 123, 136, 85} e $Y_{\tau \text{Tradicional}}$= {183, 185, 159, 185, 190, 139, 190}. A hipótese nula ($H_{0\tau}$) é $\mu_{T\tau \text{MetaUsaERPWeb}} = \mu_{T\tau \text{Tradicional}}$, ou seja, os valores médios para os dois processos são iguais. Para rejeição de $H_{0\tau}$ a favor das hipóteses alternativas tem-se que:

- $H_{1\tau}$: $\mu_{T\tau \text{MetaUsaERPWeb}} < \mu_{T\tau \text{Tradicional}}$
- $H_{2\tau}$: $\mu_{T\tau \text{MetaUsaERPWeb}} > \mu_{T\tau \text{Tradicional}}$

Utilizando-se o Teste U de Mann–Whitney tem-se que $U_{\tau \text{MetaUsaERPWeb}}$= 1 + 2 + 3 + 4 + 5 + 6 + 8 = 29 e $U_{\tau \text{Tradicional}}$= 7 + 9 + 10 + 11 + 12 + 13 + 14 = 76; logo W = $U_{\tau \text{MetaUsaERPWeb}}$. É possível verificar, através da tabela estatística do Teste U de Mann-Whitney, que o P-valor = 0,0016. Assim é possível rejeitar a hipótese nula com 5% de significância, para o Teste U de Mann–Whitney.

Para o teste-T utilizou-se as Equações 4.4 e 4.5 para cálculo de $t_{0\tau}$ as variáveis são substituidas conforme se segue:

- $\overline{x} = \mu_{T\tau \text{MetaUsaERPWeb}}$
- $\overline{y} = \mu_{T\tau \text{Tradicional}}$
- $S_X{}^2 = S_{X_{\tau \text{MetaUsaERPWeb}}}{}^2$
- $S_Y{}^2 = S_{Y_{\tau \text{Tradicional}}}{}^2$
- n = $|X_{\tau \text{MetaUsaERPWeb}}|$
- m = $|Y_{\tau \text{Tradicional}}|$

Com base nas amostras de τ, tem-se que $|X_{\tau \text{MetaUsaERPWeb}}| = |Y_{\tau \text{Tradicional}}| = 7$ e que os valores médios são $\mu_{T\tau \text{MetaUsaERPWeb}}$= 109,43 e $\mu_{T\tau \text{Tradicional}}$= 175,86. Logo obtêm-se que $S_{X_{\tau \text{MetaUsaERPWeb}}}{}^2 = 583{,}62$ e $S_{Y_{\tau \text{Tradicional}}}{}^2 = 376{,}81$. Portanto calcula-se $S_{P\tau} = 21{,}91$ e $t_{0\tau}$ = -5,67.

O número de graus de liberdade é $gl_\tau = |X_{\tau \text{MetaUsaERPWeb}}| + |Y_{\tau \text{Tradicional}}| - 2 = 7 + 7 - 2 = 12$. Na tabela padrão de distribuição de probabilidade estatística t de Student verifica-se que $t_{0{,}00015} = 5{,}44$. Como $|t_{0\mu_P}| > t_{0{,}00015}$ rejeita-se a hipótese nula com 0,015% de significância.

A segunda etapa consiste na análise da produtividade das equipes no experimento, tem-se como entrada duas amostras independentes: $X_{\phi \text{MetaUsaERPWeb}}$= {2662, 2643, 2640, 2529, 2524, 2511, 2571} e $Y_{\phi \text{Tradicional}}$= {1945, 2026, 2214, 1925, 1960, 2068, 2040}. A hipótese nula ($H_{0\phi}$) é $\mu_{P\phi \text{MetaUsaERPWeb}} = \mu_{P\phi \text{Tradicional}}$, ou seja, os valores médios para os dois processos são iguais. Para rejeição de $H_{0\phi}$ a favor das hipóteses alternativas tem-se que:

- $H_{1\phi}$: $\mu_{P\phi \text{MetaUsaERPWeb}} < \mu_{P\phi \text{Tradicional}}$
- $H_{2\phi}$: $\mu_{P\phi \text{MetaUsaERPWeb}} > \mu_{P\phi \text{Tradicional}}$

Com o Teste U de Mann–Whitney calcula-se $U_{\phi \text{MetaUsaERPWeb}} = 8 + 9 + 10 + 11 + 12 + 13 + 14 = 77$ e $U_{\phi \text{Tradicional}} = 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$; logo $W = U_{\phi \text{MetaUsaERPWeb}}$. É possível observer, através da tabela estatística do Teste U de Mann-Whitney, que o P-valor = 0,0002. Assim é possível rejeitar a hipóteses nula com 5% de significância, para o Teste U de Mann–Whitney.

De forma semelhante a etapa anterior, no Teste-T, para cálculo de $t_{0\phi}$, utiliza-se as Equações 4.4 e 4.5 substituindo as variáveis:

- $\overline{x} = \mu_{\text{P}\phi\text{MetaUsaERPWeb}}$
- $\overline{y} = \mu_{\text{P}\phi\text{Tradicional}}$
- $S_X{}^2 = S_{X_{\phi\text{MetaUsaERPWeb}}}{}^2$
- $S_Y{}^2 = S_{Y_{\phi\text{Tradicional}}}{}^2$
- $n = |X_{\phi\text{MetaUsaERPWeb}}|$
- $m = |Y_{\phi\text{Tradicional}}|$

Com base nas amostras de ϕ, tem-se que $|X_{\phi\text{MetaUsaERPWeb}}| = |Y_{\phi\text{Tradicional}}| = 7$ e que os valores médios são $\mu_{\text{P}\phi\text{MetaUsaERPWeb}} = 2582{,}86$ e $\mu_{\text{P}\phi\text{Tradicional}} = 2025{,}43$. Logo obtêm-se que $S_{X_{\phi\text{MetaUsaERPWeb}}}{}^2 = 4135{,}81$ e $S_{Y_{\phi\text{Tradicional}}}{}^2 = 9736{,}62$. Portanto calcula-se $S_{\text{P}\phi} = 83{,}28$ e $t_{0\phi} = 12{,}52$.

O número de graus de liberdade é $gl_\phi = |X_{\tau\text{MetaUsaERPWeb}}| + |Y_{\tau\text{Tradicional}}| - 2 = 7 + 7 - 2 = 12$. Na tabela padrão de distribuição de probabilidade estatística t de Student verifica-se que $t_{0{,}000000035} = 12{,}35$. Como $|t_{0\phi}| > t_{0{,}000000035}$ rejeita-se a hipótese nula com 0,0000035% de significância.

A análise de dados foi realizada utilizando o *software* Microsoft Excel [2], o suplemento Action[3] e o *software* Minitab[4].

Tendo em vista que em ambas as etapas a hipótese nula foi rejeitada é possível tirar conclusões a respeito da influência das variáveis independentes sobre a variável dependente considerando que o experimento é válido. Como $HExp1_0$ é rejeitada então é possível afirmar que as diferenças observadas na eficiência dos grupos pares e grupos ímpares possuem significância estatística. Concluí-se, portanto, que provavelmente a mudança na eficiência dos grupos foi devido aos processos usados como tratamento e não um acaso por erros aleatórios de amostragem.

Observando-se as Tabelas 4.3 e 4.5 verifica-se que $\mu_{\text{TTradicional}} = 184{,}4$ e $\mu_{\text{TMetaUsaERPWeb}} = 109{,}4$ assim como $\mu_{\text{PTradicional}} = 701{,}49$ e $\mu_{\text{PMetaUsaERPWeb}} = 1.477{,}90$. Logo, $\mu_{\text{TMetaUsaERPWeb}} < \mu_{\text{TTradicional}}$ e $\mu_{\text{PMetaUsaERPWeb}} > \mu_{\text{PTradicional}}$ satisfazem a hipótese $HExp1_1$ validando-a em detrimento da hipótese H_2. Assim existem indícios para se afirmar que equipes utilizando o MetaUsaERPWeb são, em geral, mais eficientes do que equipes utilizando a abordagem tradicional.

[2] http://office.microsoft.com/pt-br/excel/
[3] http://www.portalaction.com.br/
[4] http://www.minitab.com/

Embora não tenham sido declaradas formalmente esse resultado está de acordo com as expectativas iniciais do experimento onde os artefatos construídos na engenharia de domínio tornariam mais ágeis os engenheiros de aplicação.

Por fim, é importante manter em mente que as conclusões a respeito desse trabalho são válidas para um experimento realizado sob condições controladas restringindo-se ao escopo de desenvolvedores de software em ambiente acadêmico onde o estudo de caso foi executado. Por questões de validade, visando expandir a generalização do fenômeno observado para um contexto mais amplo, é necessário que novos estudos sob diferentes condições sejam realizados também comparando o uso do MetaUsaERPWeb ao uso de outras abordagens de desenvolvimento, tais como métodos ágeis. Isso permitirá uma validação mais abrangente das hipóteses de pesquisa.

4.3.2 Análise de requisitos não funcionais de qualidade

A fim de verificar a qualidade da estrutura interna do código produzido através do estudo de caso, alguns grupos foram selecionados e uma análise automática foi executada resultando em alguns índices.

Para esta análise tem-se a seguinte questão de pesquisa: *há diferença na qualidade das aplicações geradas pelas equipes MetaUsaERPWeb e tradicional utilizando as métricas CA e LCOM?* São levantadas as seguintes hipóteses

- Hipótese nula ($HExp2_0$): Não há diferença significativa entre as equipes MetaUsaERPWeb e tradicional em nenhuma das métricas;
- Hipótese alternativa ($HExp2_1$): As equipes MetaUsaERPWeb possuem ambas métricas significativamente melhores em relação as equipes tradicionais;
- Hipótese alternativa ($HExp2_2$): As equipes tradicionais possuem ambas métricas significativamente melhores em relação as equipes MetaUsaERPWeb;
- Hipótese alternativa ($HExp2_3$): As equipes MetaUsaERPWeb possuem a métrica CA significativamente melhor em relação as equipes tradicionais e as equipes tradicionais possuem a métrica LCOM significativamente melhor em relação as equipes MetaUsaERPWeb;
- Hipótese alternativa ($HExp2_4$): As equipes MetaUsaERPWeb possuem a métrica LCOM significativamente melhor em relação as equipes tradicionais e as equipes tradicionais possuem a métrica CA significativamente melhor em relação as equipes MetaUsaERPWeb;

Quatro grupos foram selecionados do grupo original de doze. Estes foram escolhidos pois produziram os melhores resultados. Os grupos selecionados, que utilizaram o MetaUsaERPWeb, foram 2 e 12; já os grupos 3 e 11 foram escolhidos para representar os que desenvolveram de modo tradicional.

O interesse desta análise está em duas metricas produzidas pelo analisador PhpMetrics[5]: CA (*Afferent Coupling*) e LCOM (*Lack of Cohesion of Methods*) (CHIDAMBER; KEMERER, 1994; LI; HENRY, 1993).

[5]http://www.phpmetrics.org/

A primeira verifica qual é o nível de acoplamento das classes entre si, ou seja, o quanto uma classe depende da outra. Do ponto de vista da manutenção quanto menor o acoplamento melhor pois menor a chance da alteração em uma parte da aplicação influenciar outro ponto; em outras palavras, menor a chance da alteração em um classe impactar o funcionamento da outra. Dessa forma é seguro apenas acopar-se a classes estáveis, ou seja, que não será alterada no futuro (ou é pouco provável de ser alterada). Quanto maior o índice desta métrica mais acoplada a aplicação está (as classes que a compõem estão altamente acopladas).

Já a segunda refere-se a coesão da aplicação, onde uma classe é coesa se for enxuta, ou seja, possui poucas resposabilidades. Em geral, pode-se dizer que uma classe é coesa se possui apenas uma resposabilidade, já uma classe pouco coesa possui muitas responsabilidades. Esta métrica mede a falta de coesão, ou seja, quanto maior o seu índice menos coesa é a aplicação (menos coesas são as classes que a compõe).

A análise foi realizada em uma pasta específica da aplicação, a pasta *app*. Dentro da *framework* muitos arquivos são idênticos pois fazem parte do seu *core* e não são alteradas pela aplicação. Os participantes do estudo de caso alteraram apenas os arquivos desta pasta (apesar de exitirem arquivos da *framework* que não foram alteados). Esta decisão foi tomada a fim de diminuir o impacto da implementação da *framework* no desenvolvimento das aplicações específicas.

A Tabela 4.7 apresenta os índices dessas métricas. É possível observar que a média dos grupos que utilizaram o MetaUsaERPWeb é 0,24 e 1,04 para as métricas CA e LCOM, respectivamente; já os grupos que desenvolveram as aplicações utilizando a abordagem tradicional obtiveram média de 0 e 0,68 respectivamente.

Tabela 4.7: Índices das métricas CA e LCOM

Métrica	Grupo 2	Grupo 12	Grupo 3	Grupo 11
CA	0,11	0,13	0	0
LCOM	1,15	0,92	0,68	0,68

Este resultado não é surpreendente uma vez que a implementação dos grupos ímpares mostrou-se mais simples e menos organizada (em termos de MVC) logo o código desenvolvido apresenta essa falta de complexidade e este fato impacta nesta análise. Também é esperado que o código produzido a partir de um transformador automático possua alguns vícios e este se extenda por toda a aplicação; estes vícios devem ser resolvidos em melhorias futuras deste trabalho a fim de melhorar a arquitetura interna da aplicação e sua manutenção. Desta forma a HExp2_0 é rejeitada satisfazendo a HExp2_2.

4.3.3 Análise da aceitação da proposta

Para identificar a avaliação dos participantes ao final do estudo foi aplicado um formulário de avaliação, baseado no modelo TAM (*Technology Acceptance Model*) (DAVIS, 1989) que tem como objetivo coletar a aceitação e o uso da técnica. Cada grupo recebeu o formulário adequado à abordagem utilizada, porém, em essência, as informações solicitadas eram as mesmas.

A questão de pesquisa desta análise é a seguinte: *a abordagem MetaUsaERPWeb é aceita pela maioria dos participantes do estudo de caso?* Nesta análise não cabe hipóteses.

Os grupos tradicionais relataram que não utilizaram nenhum tipo de ferramenta para geração de código para os módulos do ERP, exceto a utilização do CakePHP que era obrigatória.

Quando questionados sobre a dificuldade de execução de cada uma das tarefas 60% dos participantes dos grupos ímpares tiveram pouca (32%) ou média (28%) dificuldade na execução do planejamento já a maioria dos participantes dos grupos pares (67%) afirmaram que tiveram pouca dificuldade nessa fase, o único que julgou ter tido muita dificuldade nessa fase justificou com o grau de complexidade existente na definição das entidades relacionadas. Na tarefa de implementação 84% dos participantes dos grupos tradicionais tiveram muita dificuldade na sua execução enquanto a maioria os integrantes dos grupos MetaUsaERPWeb não tiveram dificuldade (44%) ou tiveram pouca dificuldade (33%) nessa mesma tarefa. Por fim 44% dos participantes dos grupos ímpares experienciaram muita dificuldade nos testes da aplicação, por outro lado 70% dos outros participantes tiveram pouca dificuldade nessa etapa. A Tabela 4.8 e a Figura 4.12 apresentam a distribuição das dificuldades relatadas pelos integrantes dos grupos.

Tabela 4.8: Dificuldade na execução das tarefas

	Projeto		Implementação		Teste	
Dificuldade	Trad.	MEW.	Trad.	MEW.	Trad.	MEW.
Muita dificuldade	12%	4%	84%	4%	44%	0%
Dificuldade média	28%	22%	16%	19%	16%	4%
Pouca dificuldade	32%	67%	0%	33%	20%	70%
Nenhuma dificuldade	28%	7%	0%	44%	20%	26%

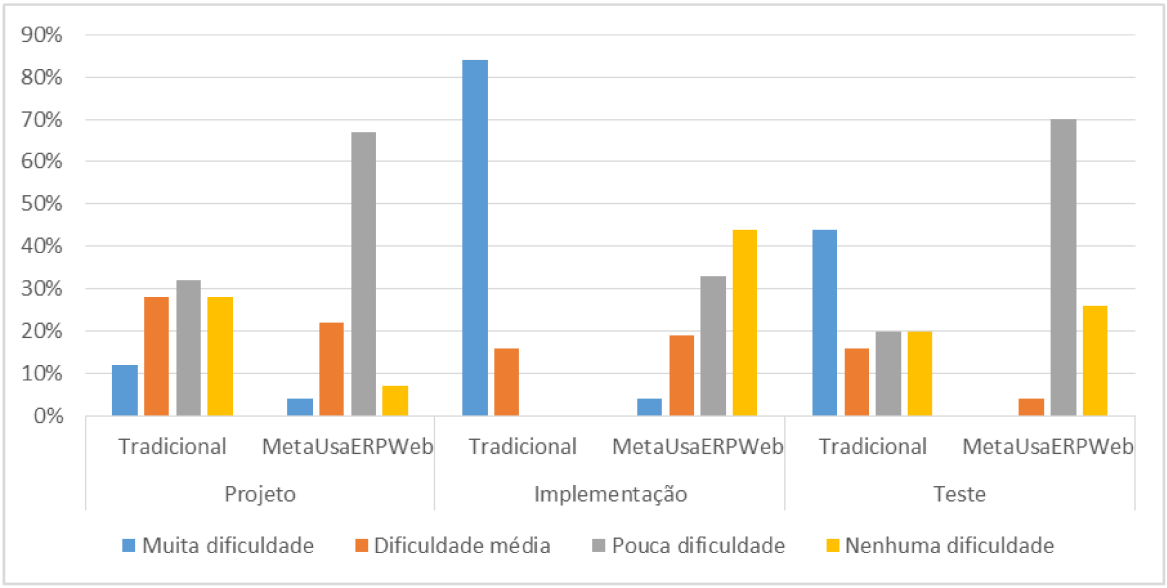

Figura 4.12: Gráfico de dificuldade na execução por tarefas

Os grupos que utilizaram a abordagem tradicional foram questionados sobre em quais momentos as heurísticas de usabilidade foram lembradas e aplicadas durante a execução de cada

etapa do estudo de caso: 56% não lembraram das técnicas durante as fases de projeto e implementação, já 80% não se atentaram para as heurísticas durante os testes da aplicação. Esse resultado não é surpreendente já que é esperado maior esforço no funcionamento da aplicação. A Figura 4.13 demonstra a distribuição do uso das heurísticas de usabilidade de acordo com as etapas do estudo de caso dos grupos ímpares.

Tabela 4.9: Aplicação das heurísticas de usabilidade

Uso	Projeto	Implementação	Teste
Usei as principais e mais importantes	4%	0%	0%
Usei apenas algumas	40%	44%	20%
Não me lembrei	56%	56%	80%

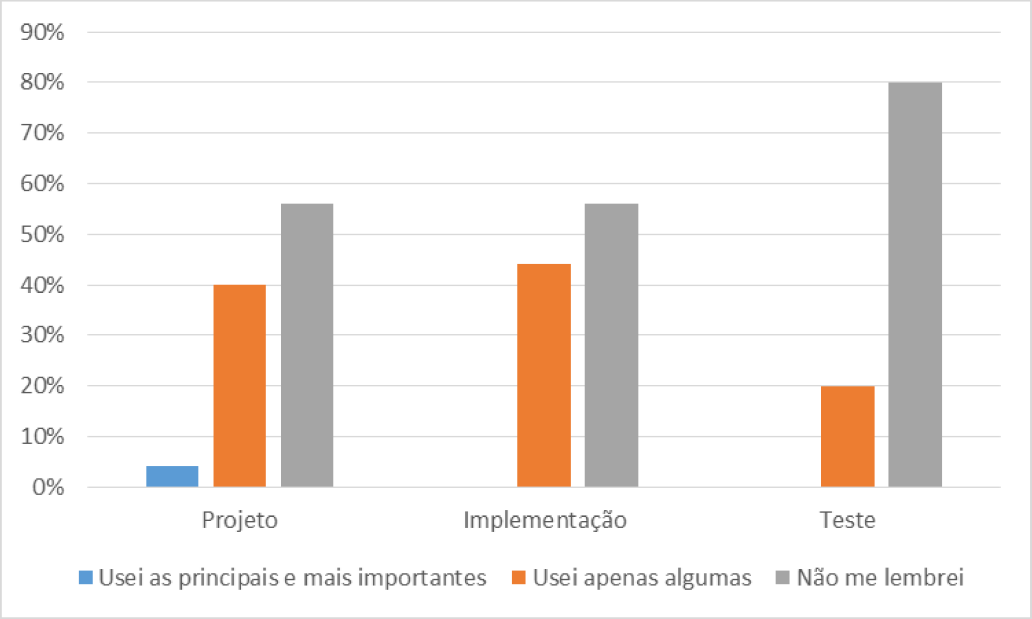

Figura 4.13: Gráfico de aplicação das heurísticas de usabilidade

Enquanto a maioria dos participantes sequer lembraram de aplicar as heurísticas de usabilidade 94% dos integrantes que utilizaram a abordagem MetaUsaERPWeb afirmaram que não precisaram lembrar dessas técnicas para serem aplicadas automaticamente pelo transformador de código apresentado nesse trabalho.

Os integrantes dos grupos ímpares ainda foram perguntados se considerariam que a existência de um processo que forneça um roteiro e mecanismos especificamente voltados para o desenvolvimento de módulos ERP facilitariam a tarefa de desenvolvimento recém completada. Nessa questão (Figura 4.14) 84% dos participantes concordaram totalmente que esse processo auxiliaria o desenvolvimento, 12% concordaram amplamente enquanto 4% concordaram parcialmente; nenhum participante discordou em nenhum nível.

De forma semelhante os participantes que utilizaram o MetaUsaERPWeb foram questionados se a aplicação do fluxo da metamodelagem auxiliou no desenvolvimento da aplicação onde todos os participantes concordaram, em algum nível, que o seu uso auxiliou o desenvolvimento. A

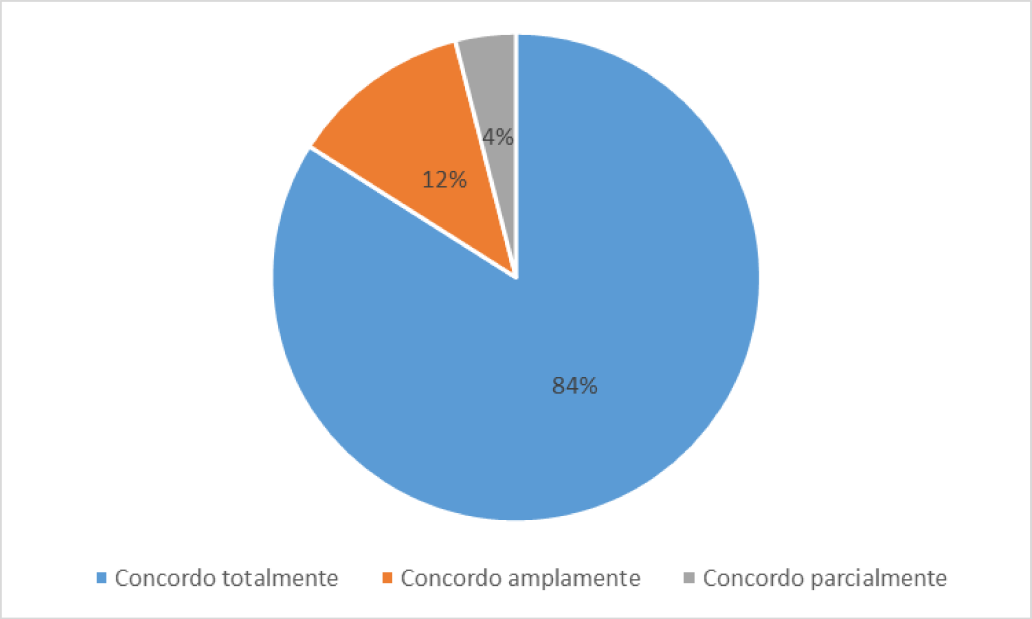

Figura 4.14: Mecanismos de ajuda para desenvolvimento de ERP

Figura 4.15 apresenta em detalhes que 85% dos integrantes que utilizaram o MetaUsaERPWeb concordaram plenamente com a afirmação que a aplicação da abordagem auxiliou no desenvolvimento dos módulos ERP enquanto 11% concordaram amplamente e apenas um integrante concordou parcialmente, pois julgou que alguns dos conceitos utilizados na técnica eram ligeiramente confusos.

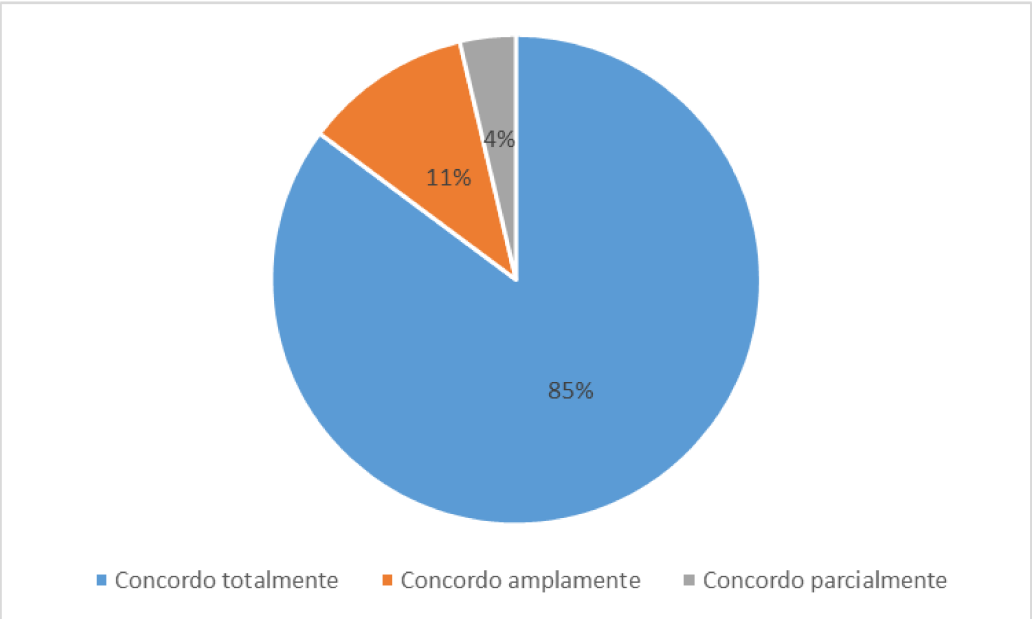

Figura 4.15: Abordagem MetaUsaERPWeb auxiliou no desenvolvimento

Ainda no questionário dos grupos pares todos os participantes concordaram, em algum nível, que foi fácil aprender a técnica, que a técnica foi usada da maneira que foi destinada, que

foi fácil ganhar habilidade no uso da técnica. Enquanto isso apenas 7% informaram que não compreendiam muito bem o que acontecia durante a interação com técnica; outros 7% discordam parcialmente que a técnica é de fácil lembrança. Por fim a Figura 4.16 apresenta que 81% dos participantes concordam totalmente que a aplicação da técnica é útil para o desenvolvimento de ERP simples enquanto os outros 19% concordam amplamente. Ainda 19% dos integrantes concordam totalmente que uso da técnica auxilia o desenvolvimente para ERP complexo, 30% concordam amplamente, 48% concordam parcialmente e apenas 3% discordaram parcialmente, a Figura 4.17 apresenta essa distribuição. Concluí-se que a maioria dos participantes aceitam a abordagem MetaUsaERPWeb.

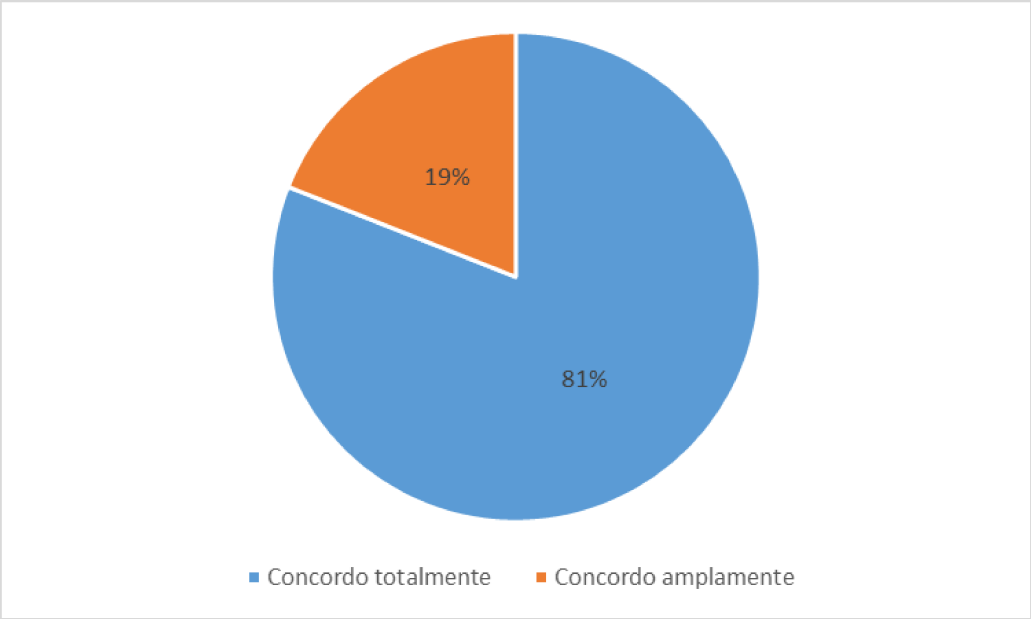

Figura 4.16: Utilidade do MetaUsaERPWeb pra desenvolvimento

Os integrantes dos grupos que utilizaram o MetaUsaERPWeb reportaram alguns erros no transformador de código como falha em validação específica no navegador Mozilla Firefox[6] ou erro na verificação das regras em alguns casos específicos. Todos os comentários foram analisados e corrigidos em versões posteriores do transformador.

4.3.4 Análise heurística de usabilidade

A análise (ou avaliação) heurística é um método utilizado para encontrar problemas de usabilidade em interfaces gráficas. Especificamente envolve avaliadores examinando a interface e avaliando sua qualidade de acordo com princípios de usabilidade, as heurísticas (NIELSEN, 1994; NIELSEN; MOLICH, 1990).

Como os objetivos desse trabalho visam as questões funcionais de usabilidade foi realizada uma análise para verificar se as heurísticas abordadas na proposta do MetaUsaERPWeb foram empregadas.

[6]https://www.mozilla.org/pt-BR/firefox/new/

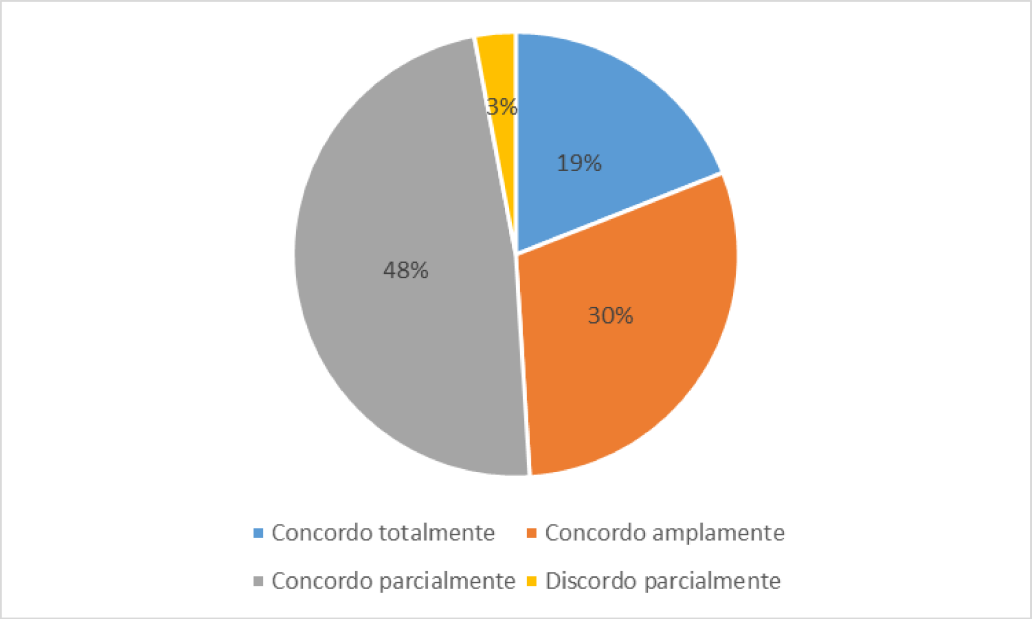

Figura 4.17: Aceitação do MetaUsaERPWeb

A questão de pesquisa desta análise é *há diferença significativa (maior ou igual a 10%) entre a média de heurísticas violadas pelas equipes MetaUsaERPWeb e tradicionais?*

- Hipótese nula ($HExp4_0$): Não há diferença significativa (maior ou igual a 10%) na média das heurísticas entre as equipes MetaUsaERPWeb e tradicional;
- Hipótese alternativa ($HExp4_1$): As equipes MetaUsaERPWeb violaram significativamente (maior ou igual a 10%) menos heurísticas do que as equipes tradicionais;
- Hipótese alternativa ($HExp4_2$): As equipes tradicionais violaram significativamente (maior ou igual a 10%) menos heurísticas do que as equipes tradicionais.

O material usado para a análise foram as aplicações geradas pelos grupos pares e ímpares no estudo de caso. Visando uma análise mais detalhada das 14 aplicações resultantes 4 foram selecionadas para a avaliação, sendo duas geradas a partir da abordagem MetaUsaERPWeb e duas a partir da abordagem tradicional. A escolha das quatro aplicações foi feita após todas serem estudadas e avaliadas, sendo consideradas as implementações mais completas.

Os grupos foram denominados de forma diferente da denominação no estudo de caso. A Tabela 4.10 apresenta o mapeamento dos grupos.

Tabela 4.10: Grupos selecionados para a análise heurística

Grupo selecionado	Identificação na análise heurística
Grupo 2	Grupo A
Grupo 12	Grupo B
Grupo 3	Grupo C
Grupo 11	Grupo D

A análise heurística foi executada por alunos de graduação e mestrado em Ciência da Computação que cursavam a disciplina de Interface Humano-Computador da Universidade Federal de São Carlos - *campus* Sorocaba em setembro de 2014. Antes da análise os alunos tiveram aulas sobre os princípios heurísticos e suas aplicações e sobre as heurísticas de Nielsen.

Os alunos responderam um pré-questionário (Apêndice J) para coletar seu conhecimento sobre inspeção heurística. Os 18 alunos foram divididos entre as quatro aplicações da seguinte forma: 5 alunos avaliaram o grupo A, 4 alunos avaliaram o grupo B, 4 alunos valiaram o grupo C e 5 alunos avaliaram o grupo D.

A análise heurística é orientada a tarefas que são executadas pelos avaliadores, durante essas tarefas as violações de qualidade, baseadas nas heurísticas de Nielsen, são registradas. O registro se faz identificando qual tarefa está sendo avaliada e quais heurísticas foram violadas.

Nessa avaliação oito tarefas foram propostas (Apêndice K) contendo ações mais simples como criação de produtos e vendas, passando pelo gerenciamento (edição, cancelamento, etc) desses itens até situações críticas de erros e validações como estoque insuficiente e venda de quantidade negativa de produto.

Foram disponibilizadas URLs para os alunos acessarem as aplicações já preparadas para a avaliação. Apesar da mesma aplicação ter sido avaliada por mais de um inspetor cada aluno possuia um banco de dados segmentado de forma que as ações dos outros avaliadores não interferissem na sua análise.

Resultados estatísticos

Para a análise de resultados utilizou-se as seguintes métricas:

- ν: número de violações encontradas pelos avaliadores;
- β: número de avaliadores atribuídos a um determinado grupo;
- μ_{V}: número médio de violações encontradas de um determinado grupo por avaliador.

Considerando que existem as equipes denominadas Y e Z pode-se dizer que a equipe Y é mais eficiente que a equipe Z se $\nu_{\mathrm{Y}} < \nu_{\mathrm{Z}}$ e $\mu_{\mathrm{VY}} < \mu_{\mathrm{VZ}}$. Também afirma-se que não há diferença entre as equipes A e B se $\nu_{\mathrm{Y}} = \nu_{\mathrm{Z}}$ e $\mu_{\mathrm{VY}} = \mu_{\mathrm{VZ}}$. A comparação com νé verdadeira apenas caso $\beta_{\mathrm{Y}} = \beta_{\mathrm{Z}}$, porém essa comparação se faz desnecessária já que a média é suficiente para essa conclusão.

Durante a análise os avaliadores relataram, em alguns casos, que o mesmo problema poderia violar mais de uma heurística e estar presente em mais de uma tarefa; nesse caso o problema relatado resultou em várias violações uma para cada combinação de heurística e tarefa. Por exemplo, se um problema viola duas heurísticas e foi identificado em três tarefas então seis violações foram contabilizadas.

Em um primeiro momento foram identificadas 769 violações em 311 problemas por todos os avaliadores. Duas análises posteriores foram realizadas por especialistas, sendo a primeira em busca de falsos positivos. Alguns problemas relatados não estavam definidos no escopo; e algumas ocorrências, por exemplo, relataram que no cadastro da venda não existia a opção de informar cliente, porém essa funcionalidade não existia no escopo do estudo de caso. Em outros

casos a descrição dos problemas estava confusa e não possuia justificativa além de ocorrências que não se enquadravam em problemas de heurísticas. Além disso, também foi analisado se o problema de fato violava a heurística apontada, caso negativo o falso positivo era assinalado.

Dessa forma foram identificadas 409 falsos positivos. Ao final tem-se 360 violações em 154 problemas sendo 65 violações do grupo A, 60 do grupo B, 110 do grupo C e 125 do grupo D. Como cada grupo foi avaliado por um número diferente de avaliadores é natural alguns possuirem mais violações. Para uma análise mais assertiva temos que cada avaliador do grupo A encontrou, em média, 13 violações; cada avaliador do grupo B encontrou 15 violações na média; já o grupo C teve 27,5 violações, em média, por avaliador; por fim, 25 violações foram encontradas por cada avaliador, na média, do grupo D. A Tabela 4.11 e a Figura 4.18 detalham esses dados.

Tabela 4.11: Violações encontradas por avaliador

Grupo	Avaliador	Violações encontradas	Média	Total
A	A2	24	13	65
	A3	14		
	A4	9		
	A5	8		
	A6	10		
B	B2	15	15	60
	B3	12		
	B4	20		
	B5	13		
C	C2	45	27,5	110
	C3	25		
	C4	32		
	C5	8		
D	D1	31	25	125
	D2	25		
	D3	46		
	D4	13		
	D5	10		

Com base nessas informações observa-se que as aplicações que foram geradas a partir do MetaUsaERPWeb tiveram, em média, 13,9 violações encontradas por avaliador em um total de 125 violações; já as aplicações criadas utilizando-se o método tradicional totalizaram 235 violações, registrando 26,1 violações por avaliador. Ou seja, na média, as aplicações do último caso apresentaram 88% mais violações do que as do primeiro caso.

Assim, tem-se que $\mu_{\mathrm{VA}} < \mu_{\mathrm{VB}} < \mu_{\mathrm{VD}} < \mu_{\mathrm{VC}}$, logo conclui-se que a aplicação do Grupo A é a mais eficiente e que a aplicação do Grupo C é a menos eficiente.

A Figura 4.19 apresenta os pontos da amostra das violações heurísticas distribuídos pela reta normal; é possível perceber que os pontos estão distribuídos de forma aleatória pela reta apresentando indícios que a amostra não possui distrubuição normal. Isto é confirmado uma vez que a Estatística Shapiro-Wilk = 0,85709 e o P-valor = 0,0109; logo não é possível afirmar

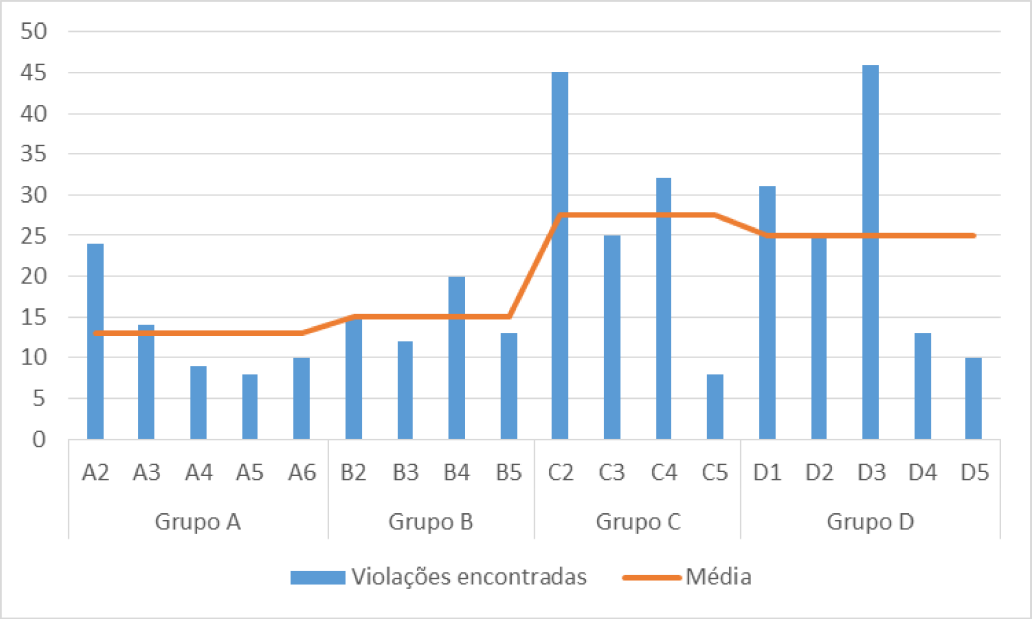

Figura 4.18: Gráfico das violações encontradas por avaliador

com 5% de significância que a amostra possui distrubuição normal.

Deseja-se comparar, porém, a eficiência com base na origem das aplicações então afirma-se que $\nu_{\text{MetaUsaERPWeb}} = \nu_{\text{A}} + \nu_{\text{B}}$ e $\nu_{\text{Tradicional}} = \nu_{\text{C}} + \nu_{\text{D}}$. Logo $\nu_{\text{MetaUsaERPWeb}} = 125$, $\nu_{\text{Tradicional}} = 235$, $\mu_{\text{VTradicional}} = 13{,}9$ e $\mu_{\text{VTradicional}} = 26{,}1$.

Como $\beta_{\text{MetaUsaERPWeb}} = \beta_{\text{Tradicional}} = 9$ então tem-se que $\nu_{\text{MetaUsaERPWeb}} < \nu_{\text{Tradicional}}$ e $\mu_{\text{VTradicional}} < \mu_{\text{VTradicional}}$, portanto conclui-se que as aplicações geradas a patir do MetaUsaERPWeb são mais eficientes que aplicações geradas do modo tradicional em relação a qualidade de interfaces. Isso porque, em média, possuem um menor número de violações.

Para verificar com algum grau de significância se é possível rejeitar a hipótese nula em favor de alguma das hipóteses alternativas aplicou-se o Teste U de Mann-Whitney. O efeito da variável dependente envolveu um aspecto, o número médio de violações encontradas (μ_{V}). Para os propósitos dessa análise utilizou-se o menor grau de significância máximo de 5%.

Tem-se como entrada duas amostras independentes: X$\mu_{\text{VTradicional}} = \{24, 14, 9, 8, 10, 15, 12, 20, 13\}$ e Y$\mu_{\text{VTradicional}} = \{45, 25, 32, 8, 31, 25, 46, 13, 10\}$. A hipótese nula ($\text{H}_0$) é $\mu_{\text{VTradicional}} = \mu_{\text{VTradicional}}$. Para rejeitação de H_0 a favor das hipóteses alternativas tem-se que:

- H_1: $\mu_{\text{VTradicional}} < \mu_{\text{VTradicional}}$
- H_2: $\mu_{\text{VTradicional}} > \mu_{\text{VTradicional}}$

Tem-se que $\text{U}_{\text{MetaUsaERPWeb}} = 1{,}5 + 3 + 4{,}5 + 6 + 7{,}5 + 9 + 10 + 11 + 12 = 64{,}5$ e $\text{U}_{\text{Tradicional}} = 1{,}5 + 3 + 4{,}5 + 6 + 7{,}5 + 13 + 14 + 15 + 16 + 17 + 18 = 106{,}5$. Logo $\text{W} = \text{U}_{\text{MetaUsaERPWeb}}$ e observando-se a tabela estatística do Teste U de Mann-Whitney o P-valor é 0,0348. Assim, através do Teste U de Mann-Whitney, rejeita-se a hipótese nula com 5% de significância.

A fim de garantir o resultado acima aplicou-se também o Teste-T. Para cálculo de t_0 utiliza-se as Equações 4.4 e 4.5 onde as variáveis são substituidas como se segue:

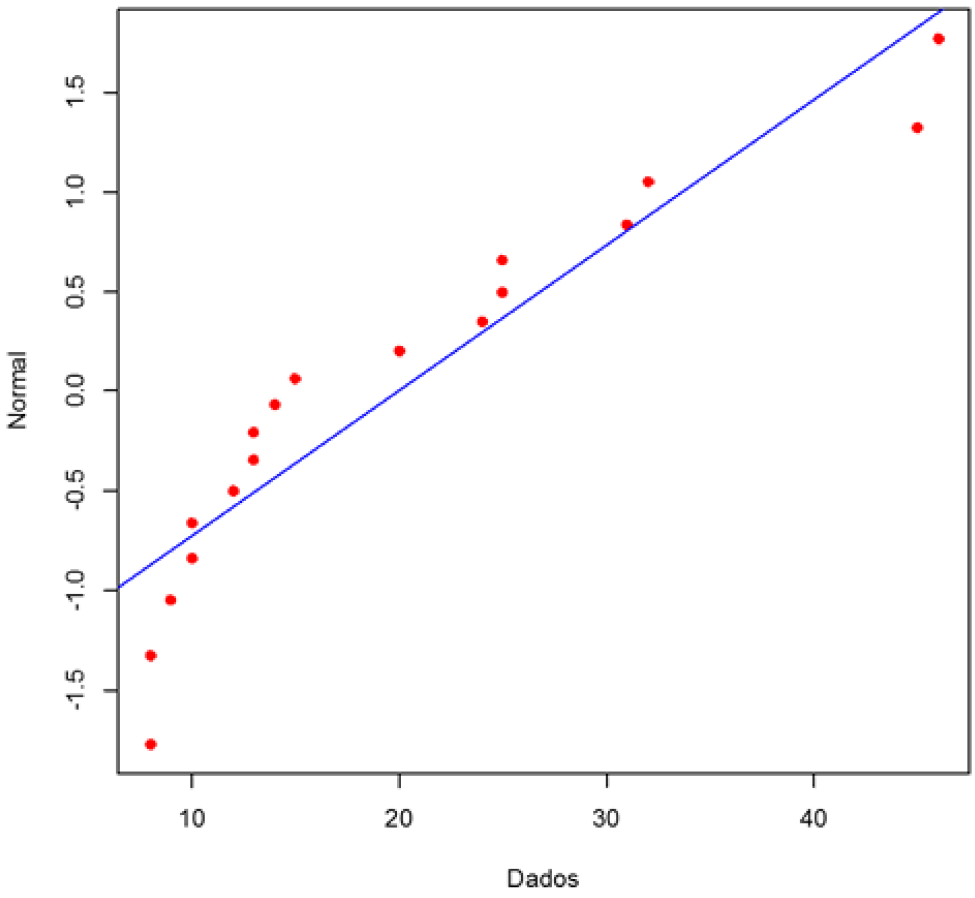

Figura 4.19: Teste de normalidade das violações heurísticas

- $\overline{x}$ = média de $\mu_{\text{V Tradicional}}$
- $\overline{y}$ = média de $\mu_{\text{V Tradicional}}$
- $S_X{}^2 = S_{X\mu_{\text{V Tradicional}}}{}^2$
- $S_Y{}^2 = S_{Y\mu_{\text{V Tradicional}}}{}^2$
- n = $|\mu_{\text{V Tradicional}}|$
- m = $|\mu_{\text{V Tradicional}}|$

Com base nas amostras de μ_V, tem-se que $|\mu_{\text{V Tradicional}}| = |\mu_{\text{V Tradicional}}| = 9$ e que os valores médios são $\overline{x} = 13{,}9$ e $\overline{y} = 26{,}1$. Portanto tem-se que $S_X = 27{,}36$ e $S_Y = 196{,}61$. Logo calcula-se $S_P = 10{,}58$ e $t_0 = -2{,}45$.

O número de graus de liberdade é $gl_{\mu_V} = |\mu_{\text{V Tradicional}}| + |\mu_{\text{V Tradicional}}| - 2 = 9 + 9 - 2 = 16$. Na tabela padrão de distribuição de probabilidade estatística t de Student verifica-se que $t_{0,003} = 2{,}38$. Como $|t_0| > t_{0,003}$ então rejeita-se a hipótese nula com 3% de significância.

A análise de dados foi realizada utilizando o *software* Microsoft Excel, o suplemento Action e o *software* Minitab.

Tendo em vista que a hipótese nula foi rejeitada é possível tirar conclusões a respeito da influência das variáveis dependentes sobre a variável independente considerando o experimento é válido. Como H_0 foi rejeitada é possível afirmar que as diferenças observadas na eficiência das aplicações geradas pelo metamodelo e pela abordagem tradicional possuem significância estatística. Sugere-se, portanto, que provavelmente a mudança na eficiência dos grupos foi devido as abordagens usadas como tratamento e não a um acaso por erros aleatórios de amostragem.

Como $\mu_{\text{VTradicional}} = 13{,}8$ e $\mu_{\text{VTradicional}} = 26{,}1$ então $\mu_{\text{VTradicional}} < \mu_{\text{VTradicional}}$satisfaz a hipótese H_1 em detrimento da hipótese H_2. Portanto, existem indícios para se afirmar que as aplicações implementadas pelo MetaUsaERPWeb são, em geral, mais eficientes em relação as aplicações geradas pela abordagem tradicional. Assim a $HExp4_0$ é rejeitada e a $HExp4_1$ é satisfeita.

Análise de heurísticas violadas

A análise das heurísticas mais violadas permite verificar em quais pontos o MetaUsaERPWeb foi mais eficiente na questão de usabilidade. A Tabela 4.12 apresenta em detalhes o número de violações por heurísticas e abordagens.

Tabela 4.12: Violações por heurísticas e abordagem

Heurística	Abordagem MetaUsaERPWeb	Abordagem Tradicional
H1	35	38
H2	10	26
H3	21	18
H4	28	25
H5	7	78
H6	4	3
H7	15	13
H8	0	1
H9	4	20
H10	1	13

Pode-se observar que a maior diferença está na H5, de prevenção de erros, isto porque o metamodelo implementa a verificação de estoque no módulo de venda evitando que o estoque torne-se negativo. Na sequência tem-se as heurísticas H2 e H9, seguida pelas heurísticas H10 e H1.

De acordo com a Subseção 3.3 este trabalho focou em cinco heurísticas: H1, H4, H5, H6 e H9; assim, segundo a Tabela 3.6 os FUFs 1 e 3 também estão considerados. De acordo com a Tabela 4.12 é possível observar que H5 e H9 tiveram queda significativa no número de violações heurísticas; a heurística H1 também apresentou leve queda nas violações.

Por fim H4 e H6 apresentaram mais violações; a maioria das violações da primeira deu-se por todo o aplicativo estar em inglês e alguns botões em português, isso aconteceu pois o transformador não definiu palavra exata para o botão deixando sob a responsabilidade do navegador, como os navegadores estavam em português essa violação foi altamente apontada.

A heurística H6 aconteceu pois o transformador preencheu todos os formulário e tabelas com títulos genéricos desencadeando assim algumas violações heurísticas.

Na subseção 3.3.1 um novo FUF foi incluído, o FUF8 a fim de garantir a consistência das informações. A heurística H5 mostra claramente que a abordagem MetaUsaERPWeb possui melhor prevenção de erros, esse fato ocorre pois o transformador gerou as regras e comportamentos necessários para garantir que a consistência de informações fosse respeitada. Além disso H9 também confirma esse fato uma vez que essas mesmas regras e comportamentos garantiram a correta recuperação de erros por parte do usuário principalmente caso estivesse próximo a sacrificar a consistência das informações por algum erro na decisão.

4.4 Ameaças à validade

Desde as primeiras fases do experimento passando pela condução com os participantes até a sua análise deve-se ter cautela a sua validade, ou seja, se os resultados obtidos são válidos para o estudo específico como para situações mais genéricas com uma população mais ampla daquela que executou o esperimento (WOHLIN, 2000). Existem quatro ameaças principais que podem colocar um experimento em risco (WOHLIN, 2000): conclusão, interna, construção e externa. Cada uma abordada a seguir.

A validade de conclusão refere-se a habilidade de concluir corretamente a respeito do efeito dos tratamentos nas variáveis dependentes como, por exemplo, escolha dos métodos estatísticos ou forma que o experimento foi executado. Tratando-se de um estudo de caso que possui duas metodologias diferentes (MetaUsaERPWeb e tradicional) foram adotados dois testes estatísticos: Teste-T e Teste U Mann-Whitney. Estes foram escolhidos pois de acordo com o teste de normalidade Shapiro-Wilk nem todas as amostras estão distribuídos de forma normal e esta condição é um pré-requisito para a utilização do Teste-T. Apesar do Teste-T ser robusto o suficiente para suportar alguns desvios nessa condição (WOHLIN, 2000) o Teste U Mann-Whitney não possui este requisito portanto ambos foram executados neste trabalho. Os dois testes apresentaram resultados semelhantes. As medidas que deviam ser feitas pelos participantes indepentem do julgamento humano e inclusive algumas foram normalizadas (número de linhas produzidas, garantindo que linhas em branco desnecessárias fossem desconsideradas, por exemplo). Por fim os grupos foram distribuidos de forma homogênea a fim de garantir que possuissem níveis de experiência similares.

A validade interna faz referência a habilidade de assegurar que os resultados foram obtidos em decorrência dos tratamentos e não por coincidência. Neste estudo de caso participaram estudantes de gradução e mestrado em Ciência da Computação que já possuem certa experiência em desenvolvimento de software conforme apresentado no formulário de caracterização; assim assume-se que sejam representativos para a população dos desenvolvedores de software. Conforme indicado no item anterior os participantes foram divididos de forma homogênea garantindo que o nível de conhecimento e experiência fossem similares. Não foi oferecido nenhum tipo de recompensa ou favorecimento aos alunos para participar do estudo de caso, assim evitando-se empenho anormal visando esses favorecimentos ou recompensas. Por fim todos os participantes estavam sob as mesmas condições durante uma única Seção e simultaneamente para a execução

do estudo de caso.

Validade de construção refere-se a habilidade de generalizar o resultado do experimento ao conceito ou teoria envolvidos no estudo. Este estudo de caso teve o objetivo comparar duas abordagens de desenvolvimento com respeito ao seu impacto em diversas métricas de eficiência e eficácia da equipe de desenvolvimento bem como na qualidade do software produzido. Assim o estudo foi condizido de modo que a fosse possível realizar de forma adequada a análise dessas métricas. Durante a executação do estudo de caso os participantes não foram informados sobre quais as métricas que seriam consideradas na análise. Assim evitou-se que os participantes se comportassem de maneira incomum para maximizar os resultados das métricas consideradas.

A última ameaça, validade externa, faz referência a habilidade de generalizar os resultados do experimento para um contexto mais amplo do que foi selecionado para o estudo. Existem três riscos principais: (1) escolha errada dos participantes; (2) conduzir o experimento em ambiente inapropriado; e (3) desempenhar o estudo em um período de tempo que afete o resultado. Os participantes do estudo de caso podem ser considerados, em geral, representativos para a população dos desenvolvedores de software, conforme citado na segunda ameaça. O experimento foi conduzido no laboratório informatizado da Universidade Federal de São Carlos - *campus* Sorocaba com equipamentos atualizados e ferramentas e tecnologia de desenvolvimento adequados para ambientes industriais. Em relação ao último ponto o estudo de caso foi executado durante uma aula com período máximo de 3 horas e 40 minutos evitando que os resultados fossem afetados em decorrência de tédio ou desgaste dos participantes.

4.5 Considerações finais

Este capítulo apresentou um estudo de caso aplicado nos alunos de graduação e mestrado em Ciência da Computação pela Universidade Federal de São Carlos - *campus* Sorocaba com o objetivo de validar a proposta desta dissertação. Um pequeno ERP de varejo foi proposto e os participantes geraram aplicações de duas formas diferentes: abordagem tradicional de desenvolvimento e utilizando o MetaUsaERPWeb. Após análise concluiu-se que os alunos que utilizaram o MetaUsaERPWeb foram mais rápidos, geraram mais linhas de código e interfaces com maior qualidade em relação as características funcionais de usabilidade (FUF).

Apesar de não ter sido declarado explicitamente acreditava-se nesse resultado já que este é o objetivo da MDD e é o objetivo específico deste trabalho. O resultado confirmou-se mesmo com integrantes menos acostumados com as tecnologias utilizadas alocados nos grupos MetaUsaERPWeb, ou seja, mesmo não tendo domínio sobre as tecnologias foram mais eficientes que os integrantes que possuiam maior domínio.

Capítulo 5

Conclusões e trabalhos futuros

Este trabalho propôs uma abordagem dirigida a modelos para o desenvolvimento de sistemas ERP de varejo com características funcionais de usabilidade. Durante as pesquisas observou-se que as aplicações dentro do domínio do ERP possuem problemas referentes a experiência do usuário. Isto ocorre pela necessidade que esses sistemas possuem de trabalhar com muitos dados e regras de negócio o esforço dos desenvolvedores acaba sendo totalmente dedicado a garantir a consistência das informações e não em formas de tornar interação do usuário mais simples e usual.

Através do estudo das heurísticas de Nielsen e dos FUFs na questão de usabilidade; e de um levantamento de funcionalidades e violações heurísticas de alguns ERP de mercado, concluiu-se que é possível automatizar algumas regras de usabilidade no desenvolvimento desse tipo de sistema considerando o estudo do domínio. A partir deste estudo foi criado um mapeamento FUF x Heurísticas de Nielsen auxiliando na visão funcional e de interação pelos projetistas além da inspeção do produto. Foi identificado, durante o estudo, um novo FUF não previsto no levantamento dos autores; este novo FUF foi proposto e utilizado durante esta dissertação.

Assim um metamodelo, denominado MetaUsaERPWeb, com o foco em ERP de varejo e em características funcionais de usabilidade foi desenvolvido. Além do metamodelo, também foi criado um transformador M2C que possui o PHP como código alvo.

Para validar a abordagem do desenvolvimento a partir do metamodelo e do transformador foi executado um estudo de caso com os alunos de graduação e mestrado de Ciência da Computação da UFSCar - *campus* Sorocaba. Neste estudo de caso os participantes foram divididos em dois grupos: um desenvolveu um ERP através da abordagem padrão de desenvolvimento, já o segundo utilizou o MetaUsaERPWeb e seu transformador para a mesma tarefa. Pode-se concluir, após diversas análises, que o grupo utilizando o MetaUsaERPWeb foi mais eficiente em relação ao tempo, número de linhas produzido e qualidade da aplicação.

Ainda em relação a validação da proposta um novo estudo de caso foi executado, desta vez com o objetivo de realizar uma análise heurística de algumas aplicações geradas a partir do estudo de caso anterior. Esta análise foi executada novamente por alunos de graduação e mestrado de Ciência da Computação da UFSCar - *campus* Sorocaba, porém não foram os mesmos participantes do primeiro estudo de caso. Após a análise verificou-se uma melhoria em 60% das heurísticas observadas.

Dentre as contribuições deste trabalho, além das já citadas, destaca-se o levantamento das principais violações de sistemas ERP de mercado onde uma análise foi executada e permitiu a sequência do projeto; de forma análoga também foi realizada a análise das principais funcionalidades do sistema do ERP de varejo de mercado. Outra contribuição deste trabalho foi a definição de um novo FUF que foca em questões altamente relacionadas a sistemas ERP: consistência das informações; este FUF não estava presente na proposta original dos autores e foi criada e utilizada neste trabalho.

A etapa de desenvolvimento possui mais três contribuições, o desenvolvimento do metamodelo, denominado MetaUsaERPWeb, para o domínio de sistemas ERP de varejo com características funcionais de usabilidade. Após este desenvolvimento foi criado o transformador M2C deste metamodelo com o PHP como plataforma alvo. Por fim a última contribuição é a validação da proposta do MetaUsaERPWeb tanto comparando-o com o desenvolvimento tradicional como em relação as características de usabilidade comparando as aplicações geradas de forma tradicional e através do MetaUsaERPWeb.

5.1 Limitações e trabalhos futuros

Foram impostas algumas limitações a este trabalho como foco apenas em sistemas ERP de varejo e cerne em apenas algumas heurísticas de Nielsen. Assim, sugere-se que esta dissertação pode permitir trabalhos futuros de pesquisa como:

- Estender a solução para outras frentes de sistemas ERP;
- Estender a solução para outras aplicações como e-commerce e redes sociais;
- Incluir outras heurísticas para automatização pela ferramenta;
- Incluir novas funcionalidades como cálculos de *sub-entities* e filtros automatizados.

5.2 Publicação e submissões

A seguinte publicação foi permitida por este trabalho: Choma, Joelma, Quintale, Diego H., Zaina, Luciana. A. M., & Beraldo, Daniela (2015). A perspective-based usability inspection for ERP systems. 17th International Conference on Enterprise Information Systems (ICEIS), pp 57-64, Barcelona, Espanha.

Além disso este trabalho será submetido ao *Journal of Software Engineering Research and Development*[1].

[1]http://www.jserd.com/

Bibliografia

BIAS, R.; MAYHEW, D. *Cost-justifying Usability: An Update for an Internet Age*. Morgan Kaufman, 2005. (Interactive Technologies Series). ISBN 9780120958115. Disponível em: <http://books.google.com.br/books?id=kDVgsGgkF4cC>.

CARROLL, J. M. Human computer interaction (hci). In: *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* [S.l.: s.n.], 2014. Disponível em: <https://www.interaction-design.org/encyclopedia/human_computer_interaction_hci.html>. Acessado: 17/02/2015.

CERNY, T.; SONG, E. A profile approach to using uml models for rich form generation. In: *Information Science and Applications (ICISA), 2010 International Conference on*. [S.l.: s.n.], 2010. p. 1–8.

CHAVARRIAGA, E.; MACÍAS, J. A. A model-driven approach to building modern semantic web-based user interfaces. *Adv. Eng. Softw.*, Elsevier Science Ltd., Oxford, UK, UK, v. 40, n. 12, p. 1329–1334, dez. 2009. ISSN 0965-9978. Disponível em: <http://dx.doi.org/10.1016/j.advengsoft.2009.01.016>.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, IEEE, v. 20, n. 6, p. 476–493, 1994.

CHITFOROUSH, F.; YAZDANDOOST, M.; RAMSIN, R. Methodology support for the model driven architecture. In: *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*. [S.l.: s.n.], 2007. p. 454–461. ISSN 1530-1362.

CIRILO, C. et al. Model driven richubi - a model-driven process to construct rich interfaces for context-sensitive ubiquitous applications. In: *Software Engineering (SBES), 2010 Brazilian Symposium on*. [S.l.: s.n.], 2010. p. 100–109.

CIRILO, C. E. et al. Experimentation of the model driven richubi process in the adaptive rich interfaces development. In: *Proceedings of the 2011 25th Brazilian Symposium on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2011. (SBES '11), p. 184–193. ISBN 978-0-7695-4603-2. Disponível em: <http://dx.doi.org/10.1109/SBES.2011.20>.

CLEMENTS, P.; NORTHROP, L. *Software Product Lines: Practices and Patterns*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. ISBN 0-201-70332-7.

CONSORTIUM, W. W. W. *Elements - HTML5*. Http://www.w3.org/TR/2010/WD-html5-20101019/elements.html. Acessado: 19/07/2015.

COOK, S. et al. *Domain-specific development with visual studio dsl tools*. [S.l.]: Pearson Education, 2007.

CZARNECKI, K. Overview of generative software development. In: *Unconventional Programming Paradigms*. [S.l.]: Springer, 2005. p. 326–341.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, JSTOR, p. 319–340, 1989.

DEITEL, P.; DEITEL, H. *Deitel® Developer Series Ajax, Rich Internet Applications, and Web Development for Programmers*. First. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008. ISBN 9780137142309.

DEMIRKOL, S. et al. Sea_l: A domain-specific language for semantic web enabled multi-agent systems. In: *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*. [S.l.: s.n.], 2012. p. 1373–1380.

DONAHUE, G. M. Usability and the bottom line. *IEEE Softw.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 18, n. 1, p. 31–37, jan. 2001. ISSN 0740-7459. Disponível em: <http://dx.doi.org/10.1109/52.903161>.

ELLEUCH, N.; KHALFALLAH, A.; AHMED, S. B. Software architecture in model driven architecture. In: *Computational Intelligence and Intelligent Informatics, 2007. ISCIII '07. International Symposium on*. [S.l.: s.n.], 2007. p. 219–223.

FRANCE, R.; RUMPE, B. Model-driven development of complex software: A research roadmap. In: *2007 Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007. (FOSE '07), p. 37–54. ISBN 0-7695-2829-5. Disponível em: <http://dx.doi.org/10.1109/FOSE.2007.14>.

GAO, J.; LI, D.; ZHENG, S. Developing real-time system based on model driven architecture. In: *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*. [S.l.: s.n.], 2006. p. 809–814.

GOSSET, W. et al. *"Student's"collected papers*. [S.l.]: Issued by the Biometrika Office, University College, 1943.

HART, A. Mann-whitney test is not just a test of medians: differences in spread can be important. *BMJ: British Medical Journal*, BMJ Group, v. 323, n. 7309, p. 391, 2001.

HIDALGA, A. de la; ZHAO, L.; SAMPAIO, P. A domain engineering approach for the development of tendering e-marketplace applications. In: *Computer Science and Software Engineering, 2008 International Conference on*. [S.l.: s.n.], 2008. v. 4, p. 927–933.

HOLSAPPLE, C. W.; WANG, Y.-M.; WU, J.-H. Empirically testing user characteristics and fitness factors in enterprise resource planning success. *Int. J. Hum. Comput. Interaction*, v. 19, n. 3, p. 325–342, 2006. Disponível em: <http://dblp.uni-trier.de/db/journals/ijhci/ijhci19.html\#HolsappleWW06>.

JESPERSEN, J. W.; LINVALD, J. Investigating user interface engineering in the model driven architecture. In: *Proceedings of the Interact 2003 Workshop on Software Engineering and HCI*. [S.l.: s.n.], 2003.

JINPING, Y.; WENJIE, S. Research on erp system based on web architecture. In: *Internet Technology and Applications (iTAP), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 1–4.

JURISTO, N.; MORENO, A.; SANCHEZ-SEGURA, M.-I. Guidelines for eliciting usability functionalities. *Software Engineering, IEEE Transactions on*, v. 33, n. 11, p. 744–758, Nov 2007. ISSN 0098-5589.

JURISTO, N.; MORENO, A. M.; SANCHEZ-SEGURA, M.-I. Analysing the impact of usability on software design. *Journal of Systems and Software*, v. 80, n. 9, p. 1506 – 1516, 2007. ISSN 0164-1212. Evaluation and Assessment in Software Engineering {EASE06}. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0164121207000088>.

JURISTO, N.; SANCHEZ-SEGURA, M.-I. *Usability Elicitation Patterns (USEPs)*. 2006. Http://www.grise.upm.es/sites/extras/2/. Acessado: 24/01/2015.

KELLY, S.; TOLVANEN, J.-P. *Domain-specific modeling: enabling full code generation*. [S.l.]: John Wiley & Sons, 2008.

LAUDON, K.; LAUDON, J. *Sistemas de informação gerenciais*. Pearson Prentice Hall, 2007. ISBN 9788576050896. Disponível em: <http://books.google.com.br/books?id=ctYoMQAACAAJ>.

LI, W.; HENRY, S. Object-oriented metrics that predict maintainability. *Journal of systems and software*, Elsevier, v. 23, n. 2, p. 111–122, 1993.

LOWGREN, J. Interaction design. In: *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* [S.l.: s.n.], 2014. Disponível em: <https://www.interaction-design.org/encyclopedia/interaction$_d$*esign.html* > .*Acessado* : 17/02/2015.

LUCRÉDIO, D. Uma abordagem orientada a modelos para reutilização de software. *Sl]: Tese de Doutorado*, 2009.

MILOSZ, M. et al. Quality improvement of erp system gui using expert method: A case study. In: *Human System Interaction (HSI), 2013 The 6th International Conference on*. [S.l.: s.n.], 2013. p. 145–152. ISSN 2158-2246.

MOGGRIDGE, B.; ATKINSON, B. *Designing interactions*. [S.l.]: MIT press Cambridge, 2007.

MONTGOMERY, D. C. *Design and Analysis of Experiments*. [S.l.]: John Wiley & Sons, 2006. ISBN 0470088109.

MUKHTAR, M. et al. Enhanced approach for developing web applications using model driven architecture. In: *Research and Innovation in Information Systems (ICRIIS), 2013 International Conference on*. [S.l.: s.n.], 2013. p. 145–150.

NIELSEN, J. *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN 0125184050.

NIELSEN, J. Heuristic evaluation. *Usability inspection methods*, v. 24, p. 413, 1994.

NIELSEN, J. *10 Usability Heuristics for User Interface Design*. 1995. Http://www.nngroup.com/articles/ten-usability-heuristics/. Acessado: 31/12/2014.

NIELSEN, J. *How to Conduct a Heuristic Evaluation*. 1995. Http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/. Acessado: 31/12/2014.

NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: ACM. *Proceedings of the SIGCHI conference on Human factors in computing systems*. [S.l.], 1990. p. 249–256.

NIELSEN, J.; NORMAN, D. *The Definition of User Experience*. 2013. Http://www.nngroup.com/articles/definition-user-experience/. Acessado: 11/07/2015.

OBRENOVIC, Z.; STARCEVIC, D. Model-driven development of user interfaces: Promises and challenges. In: *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*. [S.l.: s.n.], 2005. v. 2, p. 1259–1262.

O'REILLY, T. *What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software*. 2005. Http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html. Acessado: 17/01/2015.

OZEN, C.; BASOGLU, N. Impact of man-machine interaction factors on enterprise resource planning (erp) software design. In: *Technology Management for the Global Future, 2006. PICMET 2006*. [S.l.: s.n.], 2006. v. 5, p. 2335–2341.

PANACH, J. I.; AQUINO, N.; PASTOR Óscar. A proposal for modelling usability in a holistic {MDD} method. *Science of Computer Programming*, v. 86, n. 0, p. 74 – 88, 2014. ISSN 0167-6423. Special issue on Software Support for User Interface Description Languages (UIDL 2011). Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167642313001573>.

PANACH, J. I.; JURISTO, N.; PASTOR, O. Including functional usability features in a model-driven development method. *Computer Science & Information Systems*, v. 10, n. 3, 2013.

PANACH, J. I. et al. A framework to identify primitives that represent usability within model-driven development methods. *Information and Software Technology*, v. 58, n. 0, p. 338 – 354, 2015. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584914001566>.

PATTAL, M.; LI, Y.; ZENG, J. Web 3.0: A real personal web! more opportunities and more threats. In: *Next Generation Mobile Applications, Services and Technologies, 2009. NGMAST '09. Third International Conference on*. [S.l.: s.n.], 2009. p. 125–128.

PENDYALA, V.; SHIM, S. The web as the ubiquitous computer. *Computer*, v. 42, n. 9, p. 90–92, Sept 2009. ISSN 0018-9162.

QUIESCENTI, M. et al. Business process-oriented design of enterprise resource planning (erp) systems for small and medium enterprises. *International Journal of Production Research*, v. 44, n. 18-19, p. 3797–3811, 2006.

ROBERT, S. et al. A lightweight approach for domain-specific modeling languages design. In: *Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on*. [S.l.: s.n.], 2009. p. 155–161. ISSN 1089-6503.

SADILEK, D. A. Prototyping domain-specific language semantics. In: *Companion to the 23rd ACM SIGPLAN Conference on Object-oriented Programming Systems Languages and Applications*. New York, NY, USA: ACM, 2008. (OOPSLA Companion '08), p. 895–896. ISBN 978-1-60558-220-7. Disponível em: <http://doi.acm.org/10.1145/1449814.1449896>.

SEFFAH, A.; METZKER, E. The obstacles and myths of usability and software engineering. *Commun. ACM*, ACM, New York, NY, USA, v. 47, n. 12, p. 71–76, dez. 2004. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1035134.1035136>.

SINGH, A.; WESSON, J. Evaluation criteria for assessing the usability of erp systems. In: *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*. New York, NY, USA: ACM, 2009. (SAICSIT '09), p. 87–95. ISBN 978-1-60558-643-4. Disponível em: <http://doi.acm.org/10.1145/1632149.1632162>.

STANDARDIZATION, I. O. for. *ISO 9241-11: 1998: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) - Part 11: Guidance on Usability*. International Organization for Standardization, 1998. Disponível em: <http://books.google.com.br/books?id=TzXYZwEACAAJ>.

TORSEL, A.-M. A testing tool for web applications using a domain-specific modelling language and the nusmv model checker. In: *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*. [S.l.: s.n.], 2013. p. 383–390.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. *Introdução à engenharia de software experimental*. [S.l.]: UFRJ, 2002.

TRENNER, L.; BAWA, J. (Ed.). *The Politics of Usability: A Practical Guide to Designing Usable Systems in Industry*. 1st. ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1998. ISBN 3540761810.

TRIAS, F. Building cms-based web applications using a model-driven approach. In: *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*. [S.l.: s.n.], 2012. p. 1–6. ISSN 2151-1349.

UFLACKER, M.; BUSSE, D. Complexity in enterprise applications vs. simplicity in user experience. In: *Proceedings of the 12th International Conference on Human-computer Interaction: Applications and Services*. Berlin, Heidelberg: Springer-Verlag, 2007. (HCI'07), p. 778–787. ISBN 978-3-540-73109-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1769451.1769542>.

WAGNER, B.; MONK, E. *Enterprise Resource Planning*. 3rd. ed. Boston, MA, United States: Course Technology Press, 2008. ISBN 1423901797, 9781423901792.

WANG, G. Application of lightweight mvc-like structure in php. In: *Business Management and Electronic Information (BMEI), 2011 International Conference on*. [S.l.: s.n.], 2011. v. 2, p. 74–77.

WELIE, M. van. *Autocomplete*. 2008. Http://www.welie.com/patterns/showPattern.php?patternID=autocomplete. Acessado: 11/07/2015.

WOHLIN, C. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic, 2000. (International Series in Engineering and Computer Science). ISBN 9780792386827. Disponível em: <http://books.google.com.br/books?id=nG2UShV0wAEC>.

Apêndices

Apêndice A

Formulário de Caracterização de Participante

Este formulário tem por objetivo caracterizar sua experiência acadêmica, pessoal e profissional com relação à Ciência da Computação. Por favor, responda a TODAS as questões o mais fielmente possível. Toda informação fornecida é confidencial, sendo que seu nome e quaisquer outros meios de identificação não serão divulgados em nenhuma hipótese.

Itens marcados com um asterisco (*) são de preenchimento obrigatório.

1. **Registro acadêmico** *

_____

2. **Sua idade** *
   Marcar apenas uma oval.
   - ◯ De 18 a 24 anos
   - ◯ De 25 a 30 anos
   - ◯ De 30 a 40 anos
   - ◯ Mais que 40 anos

3. **Aluno de...** *
   Marcar apenas uma oval.
   - ◯ Graduação
   - ◯ Mestrado

4. **Ano de ingresso** *

_____

5. **Mês/Ano de conclusão** *
Ou previsão de conclusão - No "Dia" insira 01
_____
Exemplo: 15 de dezembro de 2012

6. **Experiência em linguagem de programação Java** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

7. **Experiência em linguagem de programação PHP** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

8. **Experiência na framework CakePHP** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

9. **Experiência em linguagem de marcação XHTML/HTML** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

10. **Experiência em folhas de estilo CSS** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

11. **Experiência em Unified Modeling Language (UML)** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

12. **Experiência em biblioteca JQuery** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

13. **Experiência em banco de dados MySQL** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

14. **Experiência em PHPMyAdmin** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

15. **Experiência na IDE Eclipse** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

16. **Experiência na IDE NetBeans** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

17. **Experiência em Eclipse Modeling Framework (EMF)** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

18. **Experiência em Java Emitter Templates (JET)** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

19. **Experiência no sistema de controle de versão Git** *
Assinale a opção que melhor reflita seu grau atual de experiência
Marcar apenas uma oval.

- ◯ Nenhum
- ◯ Estudei em aula ou em livro
- ◯ Pratiquei em projetos em sala de aula
- ◯ Usei em projetos pessoais ou na indústria
- ◯ Uso em grande parte dos projetos que realizo

20. **Qual das opções a seguir melhor descreve sua experiência anterior com relação ao desenvolvimento de software na prática?** *
Marcar apenas uma oval.

- ◯ Tenho desenvolvido software por conta própria (sozinho)
- ◯ Tenho desenvolvido software como membro de equipe durante cursos que realizo
- ◯ Tenho desenvolvido software como membro de equipe no mercado há menos de dois anos
- ◯ Tenho desenvolvido software como membro de equipe no mercado há mais de dois anos

21. **Quais das opções a seguir melhor descrevem sua experiência anterior com relação ao desenvolvimento de software?** *
Marque todas que se aplicam.

- ☐ Tenho experiência como gerente de projeto de software como aluno
- ☐ Tenho experiência como gerente de projeto de software no mercado
- ☐ Tenho experiência como analista de sistemas como aluno
- ☐ Tenho experiência como analista de sistemas no mercado
- ☐ Tenho experiência como desenvolvedor de software como aluno
- ☐ Tenho experiência como desenvolvedor de software no mercado

22. **Como você distribuiria o tempo gasto em sua experiência com programação?** *
A soma das escolhas deve ser 10
Marcar apenas uma oval por linha.

	0	1	2	3	4	5	6	7	8	9	10
Em esforço individual (sozinho)	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Em desenvolvimento conjunto com outras pessoas (equipe)	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯
Na supervisão de outros programadores (gerência)	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

23. **Você tem experiência em desenvolvimento web fora das aulas da faculdade?** *
Marque todas que se aplicam.

- ☐ Não, apenas desenvolvi para a Web durante a faculdade
- ☐ Sim, eu já havia desenvolvido para a Web em meu curso técnico ou outros cursos
- ☐ Sim, já desenvolvi projetos reais para a Web como freelancer
- ☐ Sim, já trabalhei em uma empresa de desenvolvimento para a Web

24. **Experiência com a técnica de Desenvolvimento Dirigido a Modelos (MDD)** *
Assinale na tabela a seguir a opção que melhor reflita seu grau atual de experiência com os item relacionado
Marcar apenas uma oval.

- ◯ Eu não tenho familiaridade com este assunto. Eu nunca fiz isto.
- ◯ Ii ou estudei sobre isto. Conheço os conceitos e/ou técnicas.
- ◯ Eu utilizo isto algumas vezes em projetos pessoais ou na indústria, mas não sou um(a) especialista.
- ◯ Eu sou muito familiar com este assunto. Eu me sentiria confortável fazendo isto.

25. **Experiência com o conceito de Interfaces Ricas para a Web** *
Assinale na tabela a seguir a opção que melhor reflita seu grau atual de experiência com os item relacionado
Marcar apenas uma oval.

- ◯ Eu não tenho familiaridade com este assunto. Eu nunca fiz isto.
- ◯ Ii ou estudei sobre isto. Conheço os conceitos e/ou técnicas.
- ◯ Eu utilizo isto algumas vezes em projetos pessoais ou na indústria, mas não sou um(a) especialista.
- ◯ Eu sou muito familiar com este assunto. Eu me sentiria confortável fazendo isto.

26. **Experiência com o conceito de Enterprise Resource Planning (ERP)** *
Assinale na tabela a seguir a opção que melhor reflita seu grau atual de experiência com os item relacionado
Marcar apenas uma oval.

- ◯ Eu não tenho familiaridade com este assunto. Eu nunca fiz isto.
- ◯ Ii ou estudei sobre isto. Conheço os conceitos e/ou técnicas.
- ◯ Eu utilizo isto algumas vezes em projetos pessoais ou na indústria, mas não sou um(a) especialista.
- ◯ Eu sou muito familiar com este assunto. Eu me sentiria confortável fazendo isto.

27. **Experiência com a técnica de Avaliação Heurística** *
Assinale na tabela a seguir a opção que melhor reflita seu grau atual de experiência com os item relacionado
Marcar apenas uma oval.

- ◯ Eu não tenho familiaridade com este assunto. Eu nunca fiz isto.
- ◯ Ii ou estudei sobre isto. Conheço os conceitos e/ou técnicas.
- ◯ Eu utilizo isto algumas vezes em projetos pessoais ou na indústria, mas não sou um(a) especialista.
- ◯ Eu sou muito familiar com este assunto. Eu me sentiria confortável fazendo isto.

Apêndice B

Cenário de Aquecimento para Estudo de Caso

B.1 Cenário

A empresa Array Enterprises está entrando no ramo de distribuição de medicamentos e está iniciando a organização de seus negócios. Como se trata de um produto perecível e controlado é obrigação da distribuidora vender apenas medicamentos próprios para o uso, ou seja, não é permitido vender unidades do produto que já tenham passado da data de validade.

Os medicamentos chegam ao armazém da empresa em lotes. Cada lote possui várias unidades de um único medicamento, todas com a mesma data de fabricação e de validade. A entrada no estoque é feita, portanto, pelas informações do lote já que necessário o controle sobre esses dados.

A Array Enterprises solicitou a criação de dois módulos de um ERP: o módulo de produto e o módulo de lote; os dois integrados visando o auxílio computacional no controle dos vencimentos dos medicamentos.

B.2 Descrição do sistema

O ERP da Array Enterprises contará com dois módulos, produto e lote, integrados. O módulo de produto contará com os dados de cada medicamento e controlará o estoque com a ajuda do módulo de lote, portanto não é permitida a entrada de material através desse módulo.

O módulo do lote contém as informações específicas do lote e se comunica com o módulo de produto gerenciando o estoque.

B.3 Requisitos do sistema

São requisitos do sistema:

R1. O sistema deverá gerenciar os produtos, contendo os seguintes campos em seu cadastro: nome, valor de venda e descrição;

R2. O sistema deverá gerenciar os lotes, contendo os seguintes campos em seu cadastro: identificação do lote, produto que o lote pertence, data de fabricação, data de validade e quantidade de produtos no lote;

R3. O sistema deverá, ao gerenciar a quantidade do lote atualizar a quantidade total de produto no estoque.

B.4 Exemplo de fluxo

Um novo lote chegou ao armazém da Array Enterprises com as seguintes informações:

- Identificação do lote: A082K
- Produto: ABC
- Data de fabricação: 12/05/2014
- Data de validade: 12/10/2014
- Quantidade: 100

Ao informar esse lote para o sistema o produto ABC (já cadastrado em um passo anterior) passa a ter 100 produtos em estoque. Em seguida chega outro lote, com as seguintes informações:

- Identificação do lote: A138D
- Produto: ABC
- Data de fabricação: 05/04/2014
- Data de validade: 05/09/2014
- Quantidade: 350

Com esse lote no sistema automaticamente o estoque do produto ABC passa a ser 450 (100 + 350). Em outro momento são retirados 20 produtos do lote A138D, ao editar o lote no sistema o estoque do produto ABC é atualizado para 430.

B.5 Observações

Apesar de não existir um campo gerenciável de estoque para o usuário no produto o sistema precisará de um campo assim para fazer o controle conforme solicita o R3.

B.6 Glossário

Gerenciar: Ações de criar, editar e excluir um registro.

Apêndice C

Cenário do Estudo de Caso

C.1 Cenário

A empresa Array Enterprises se interessou pelo ERP já entregue e fez a solicitação para outra área que atua: a revenda de produtos eletrônicos. Como não se trata de um produto perecível o controle de estoque é simples com apenas a quantidade total de cada produto.

A empresa deseja implementar o ERP no fluxo de venda. Como na política interna da empresa todos os produtos são vendidos a pronta entrega a venda só pode ser concretizada se o produto de fato existir no estoque, caso contrário a venda deve ser bloqueada.

A Array Enterprises soliciou a criação de dois módulos de um ERP: o módulo de produto e o módulo de venda; os dois integrados visando o auxílio computacional no controle de vendas com estoque disponível.

C.2 Descrição do sistema

O ERP da Array Enterprises contará com dois módulos, produto e venda, integrados. O módulo de produto contará com os dados de cada produto e controlará o estoque com a ajuda do módulo de venda, porém deve ser permitido o gerenciamento do estoque nesse módulo já que a entrada de produtos é feito diretamente por ele.

O módulo de venda contém as informações específicas da venda e se comunica com o módulo de produto dando baixa no estoque porém sem permitir que seja vendida uma quantidade maior que o estoque disponível (em outras palavras, o estoque não pode ser negativo).

C.3 Requisitos do sistema

São requisitos do sistema:

R1. O sistema deverá gerenciar os produtos, contendo os seguintes campos em seu cadastro: nome, valor de venda, quantidade em estoque e descrição;

R2. O sistema deverá gerenciar vendas, contendo os seguintes campos em seu cadastro: data da venda, produtos vendidos, quantidade de produtos vendidos e preço de venda de cada produto;

R3. O sistema deverá, ao gerenciar as vendas, atualizar o estoque automaticamente não permitindo que a venda de um produto resulte em estoque negativo.

O diagrama representado na Figura C.1 mostra os elementos esperados e as suas relações (no diagrama não estão sendo considerados os métodos, essa implementação é de responsabilidade dos grupos):

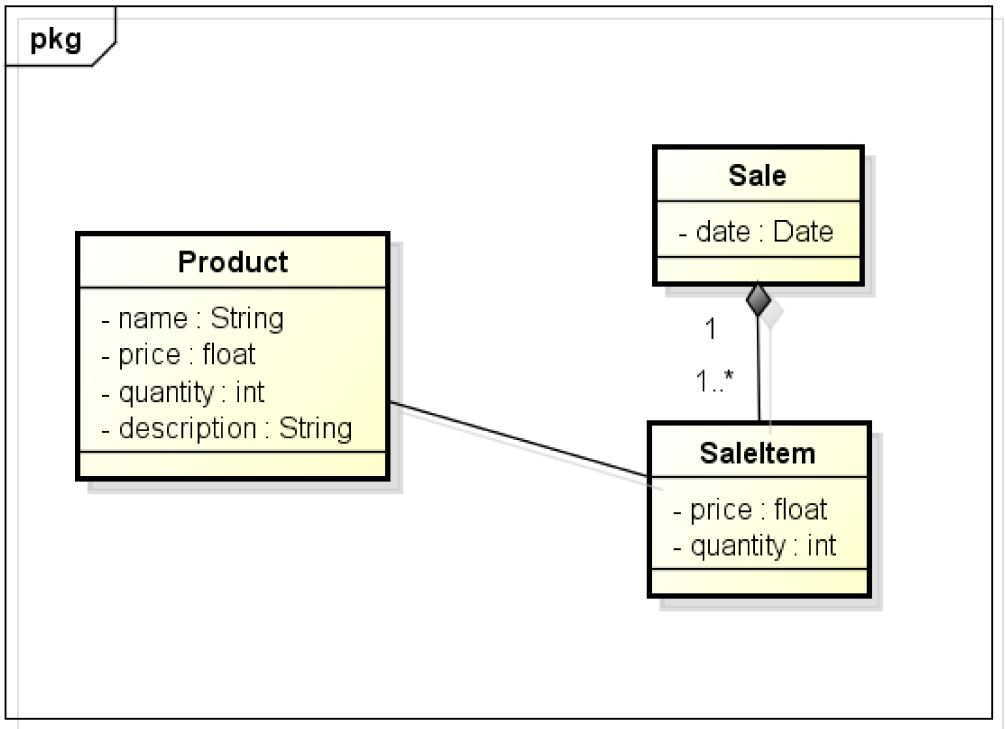

Figura C.1: Diagrama de classes da solução esperada

C.4 Exemplo de fluxo

O produto ABC já está cadastrado no sistema com 10 produtos em estoque e o produto DEF possui 200 produtos em estoque. É lançada uma venda com os seguintes dados:

- Data de venda: 12/06/2014
- Itens da venda
  - Item 1
    - Produto: ABC

    - Quantidade: 5
    - Valor de venda: 10,50
  - Item 2
    - Produto: DEF
    - Quantidade: 10
    - Valor de venda: 1,90

Assim, ao efetivar a venda, o produto ABC terá 5 produto em estoque e o produto DEF possuirá 190 em estoque. Em outro momento uma nova venda é lançada:

- Data de venda: 13/06/2014
- Itens da venda
  - Item 1
    - Produto: ABC
    - Quantidade: 10
    - Valor de venda: 12,89

O formulário dessa venda deve ser bloqueado ou o sistema deverá apresentar um aviso de erro já que se a venda fosse efetivada o estoque do produto ABC ficaria -5 o que não é permitido pelo R3.

C.5 Glossário

Gerenciar: Ações de criar, editar e excluir um registro.

Apêndice D

Material de Apoio para os Grupos Tradicionais

D.1 CakePHP

O CakePHP é uma framework escrita em PHP que utiliza a arquitetura MVC para organizar o código, também é orientada a objetos. O manual do CakePHP é encontrado em http://book.cakephp.org/2.0/en/index.html. Uma série de padrões são definidos para funcionamento correto da framework e abstração de funções:

- As tabelas no banco de dados são em minúsculo, underscored e no plural;
- Os models são em camelcase, no singular, com mesmo nome da tabela;
- Controller são em camelcase, no plural, acompanhado de "Controller";
- Views ficam dentro da pasta com nome em camelcase no plural.

Então, por exemplo:

- Tabela: products
- Model: Product
- Controller: ProductsController
- View: Products/

O CakePHP intercepta as requisições e as direciona para o controller adequado. Por exemplo, a url: http://metausaerpweb.localhost/products/index

Faz a chamada para o controller ProductsController e, dentro dessa classe, é chamada a função index(). A view /Products/index.ctp também é chamada ao final da execução da função.

Também é possível passar argumentos via URL, por exemplo: http://metausaerpweb.localhost/products

A função seria chamada assim (onde $productsController é um objeto do controller products): $productsController->add(1, teste);

A função no controller seria assim:

public function add($id, $nome) {...}

D.1.1 Model

Model é o componente que lida diretamente com os dados, cada tabela possui um model e a classe extende AppModel, os models se localizam em /app/Model. A conexão com o banco de dados é feita diretamente através desse componente (o arquivo /app/Config/database.php deve ser configurado para conexão com o banco de dados).

O model facilita as buscas no banco de dados, as configurações como ”belongsTo”, ”hasOne”e ”hasMany”fazem joins automáticos.

```
class Pessoa extends AppModel {

   var $belongsTo = array('Profissao');

}
```

Figura D.1: Exemplo de belongsTo

Então a tabela pessoas possui uma chave estrangeira com o nome profissao_id. De forma semelhante o model Profissao também está ligado a pessoas (Figura D.2).

```
class Profissao extends AppModel {

   var $hasMany = array('Pessoa');

}
```

Figura D.2: Exemplo de hasMany

Também existem os triggers que são executados automaticamente antes ou depois de um evento (Figura D.3).

```
class Pessoa extends AppModel {
  public function beforeFind($queryData) {
      $queryData['conditions'][Pessoa.ativo'] = 1;
      return $queryData;
   }
}
```

Figura D.3: Exemplo de trigger

Mais informações sobre os triggers: http://book.cakephp.org/2.0/en/models/callback-methods.html

D.1.2 Controller

O controller é o componente que une models e views, se localizam na pasta /app/Controller e extendem a classe AppController.

O controller está ligado automaticamente ao model de mesmo nome e pode ser acessado, por exemplo, para fazer a busca por um registro pelo id (Figura D.4).

```
public function view($id) {
  $conteudo = $this->Product->findById($id);
}
```

Figura D.4: Exemplo de findById

Para tabelas / models relacionados é obrigado explicitar a recusividade (Figura D.5).

```
public function view($id) {
  $this->pessoa->recursive = 1; // se for 0 nenhum join é feito
  $conteudo = $this->Pessoa->findById($id);
}
```

Figura D.5: Exemplo de recursão de findById

Pode-se salvar uma array no formato $array['nomeModel']['nomeCampo'] de forma mais rápida e simples através da função save (Figura D.6).

```
public function add() {
  if($this->request->is('post')) {
    $this->Product->create();
    if($this->Product->save($this->request->data)) {
      $this->Session->setFlash('Sucesso');
      $this->redirect(array('action' => 'index'));
    } else {
      $this->Session->setFlash('Erro');
      $this->redirect(array('action' => 'add'));
    }
  }
}
```

Figura D.6: Exemplo de save

Se a chave primária for informada e já existir esse registro no banco de dados o registro é atualizado, caso contrário é criado normalmente D.7.

```
private function upsert() {

 $product = array(
  'id' => 5, // se informado e já existir faz o update
  'name' => 'nome do produto',
  'price' => 89.50
 );
 $this->Product->create();
 $this->Product->save($product);
}
```

Figura D.7: Exemplo de upsert

Caso seja necessário salvar vários registros uma array com index pode ser criada: $array[0]['nomeModel']['nomeCampo'], $array[1]['nomeModel']['nomeCampo'],... e a função saveAll pode ser usada para salvar todos os registros.

Funções declaradas no model podem ser usadas pelo controller da mesma forma: $this->Model->nomeFuncao().

Para enviar variáveis a view é necessário explicitamente enviá-las (Figura D.8).

```
$this->set('nomeDaVariavelNaView', $variavel);
```

Figura D.8: Exemplo de set

D.1.3 View

Faz a exibição dos dados para o usuário, tem o arquivo possui a extensão .ctp e são localizados na pasta /app/View. Cada controller possui uma pasta de view e, cada função do controller possui um arquivo ctp.

A view possui tags e conteúdo do HTML e aceita comandos PHP entre as tags <?php ?>.

D.2 Código base

Esse estudo de caso alguns arquivos estão disponíveis para facilitar sua execução. Baixe esses arquivos do Moodle. O arquivo compactado possui o arquivo metausaerpweb.sql que é o banco de dados completo, também existe a pasta CakePHP que possui algumas implementações e sugestões.

O módulo de produto já está completamente implementado, o módulo de venda já possui esqueleto:

- Os models de venda e item da venda está funcional, com todas as ligações entre models feitas; outra implementações podem ser feitas;

- O controller de venda já possui esqueleto com as funções declaradas e alguns controles. Os comentários mostram onde a implementação é esperada;
- A view já possui os arquivos e esqueleto de HTML. Os comentários mostram onde a implementação é esperada.

A implementação é apenas uma sugestão, o grupo pode alterá-la se achar necessário.

Também existe um pasta no arquivo compactado, sb-admin-v2.zip, nela estão todos os elementos disponíveis do layout usado nesse estudo de caso podendo ser usado da maneira que o grupo julgar melhor.

D.3 Implementação

É recomendado no módulo de venda implementar primeiro a opção de venda de apenas um item e, na sequência, implementar para vários itens conforme o diagrama disponível no documento de cenário.

D.4 Entrega

Durante o estudo de caso um formulário deverá ser preenchido informando os passos do desenvolvimento. Esse formulário será entregue de duas formas, escrita e digital; a primeira será preenchida ao longo do exercício, a segunda após a conclusão, ambas entregas deverão ter exatamente o mesmo conteúdo.

Após o estudo de caso outro formulário deverá ser preenchido e entregue apenas de forma digital. Os arquivos gerados deverão ser entregues via Moodle. O modelo gerado (My.metaerp), a pasta do projeto CakePHP e o *dump* do banco de dados (metausaerpweb.sql), a hierarquia da pasta deverá ser:

- MetaUsaERPWeb/
  - CakePHP/
  - metausaerpweb.sql

A pasta deverá ser compactada (zip ou rar) e enviada.

Grupos que não enviarem todos os arquivos e/ou não enviarem os dois formulários caracterizarão entrega incompleta.

Apêndice E

Material de Apoio para os Grupos MetaUsaERPWeb

E.1 Metamodelo

O metamodelo completo do MetaUsaERPWeb pode ser conferido na Figura E.1.

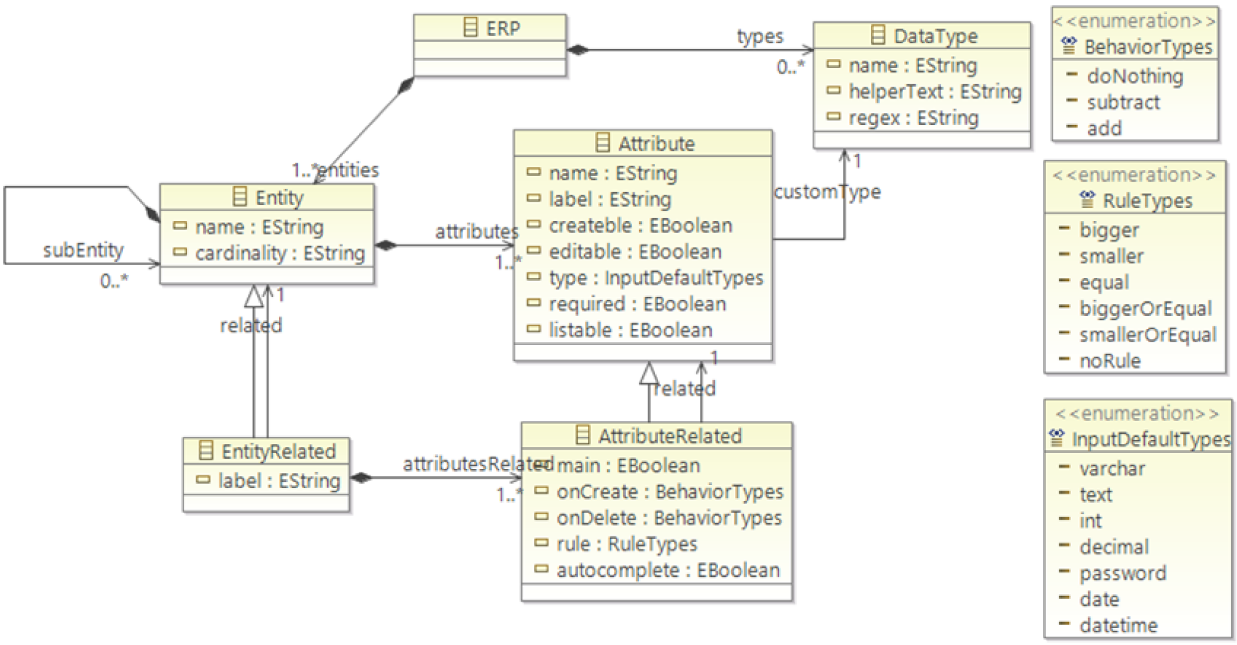

Figura E.1: MetaUsaERPWeb

O metamodelo é composto por uma série de componentes, todos explicados ao longo desse documento.

E.1.1 Entity

São as entidades que compõem o ERP onde cada módulo do ERP pode ser uma entidade, porém cada módulo pode ser composto por mais de uma entidade. Então cada entidade pode

possuir subentidades (subentidade também é uma entidade).

São campos da Entity:

- Name: o nome da entidade;
- Cadinality: cadinalidade da subentidade (em entidades esse campo é ignorado). O caractere * (asterisco) representa que não há número máximo.

E.1.2 Attribute

São os atributos de um entidade, não existe número máximo de atributos por entidade.
São campos do Attribute:

- Name: nome do atributo;
- Label: rótulo do atributo, que será exibido para o usuário;
- Creatable: define que o campo está disponível para ser alterado pelo usuário no momento da criação do registro
- Editable: define que o campo está disponível para ser alterado pelo usuário no momento da edição do registro
- Type: tipo do atributo (dos InputDefaultTypes);
- Required: define se o campo é obrigatório;
- Listable: defino se o campo será exibido na lista de registros.

E.1.3 InputDefaultTypes

O metamodelo já possui alguns tipos de dados por padrão, esses tipos são considerados os mais comuns ao lidar com sistemas ERP. Esses tipos são associados aos atributos através do campo type.

São opções do InputDefaultType:

- Varchar: sequência de 255 caracteres;
- Text: sequência maior que 255 caracteres;
- Int: número inteiro. A vírgula é usada como divisor de milhar;
- Decimal: número real. A vírgula é usada como divisor de milhar e o ponto como divisor decimal;
- Password: texto de senha;
- Date: campo de data. O padrão é o americano (MM/DD/AAAA);
- Datetime: campo de data e hora. O padrão é o americano (MM/DD/AAAA H:m:S).

E.1.4 DataType

Caso os tipos padrões não sejam suficiente é possível criar novos tipos de dados. Esse tipo pode ser associado ao Attribute.

São campos do DataType:

- Name: nome do tipo;
- HelperText: texto de ajuda exibido ao usuário;
- Regex: expressão regular que valida a entrada de dados.

E.1.5 EntityRelated

Uma entidade pode se relacionar a outra, essa relação é feita através desse componente que é filho de Entity. A relação pode ser feita apenas com uma entidade.

É campo da EntityRelated:

- Label: rótulo da entidade, que será exibido para o usuário.

E.1.6 AttributeRelated

São os atributos da EntityRelated que se relaciona aos atributos da entidade relacionada pela EntityRelated. Esse componente é filho de Attribute.

São campos do AttributeRelated:

- Main: define se é o atributo principal, que fará diretamente a ligação com a entidade relacionada;
- OnCreate: define o comportamento adotado ao criar um registro;
- OnDelete: define o comportamento adotado ao deletar um registro;
- Rule: define a regra de negócio adotada;
- Autocomplete: define se o atributo será completado automaticamente ao selecionar o atributo relacionado principal.

E.1.7 BehaviorTypes

São os comportamentos aceitos pelo metamodelo que definem o que deve acontecer ao criar ou deletar um registro de atributo relacionado.

São opções do BehaviorTypes:

- DoNothing: nada deve acontecer;
- Subtract: o atributo relacionado é subtraído da quantidade cadastrada no AttributeRelated;
- Add: o atributo relacionado é adicionado da quantidade cadastrada no AttributeRelated.

E.1.8 RuleTypes

São as regras de negócio aceitas pelo metamodelo que definem o que deve ser verificado para a regra de negócio ser respeitada.

São opções do RuleTypes:

- Bigger: o atributo relacionado deve ser maior que a quantidade no AttributeRelated;
- Smaller: o atributo relacionado deve ser menor que a quantidade no AttributeRelated;
- Equal: o atributo relacionado deve ser igual que a quantidade no AttributeRelated;
- BiggerOrEqual: o atributo relacionado deve ser maior ou igual que a quantidade no AttributeRelated;
- SmallerOrEqual: o atributo relacionado deve ser menor ou igual que a quantidade no AttributeRelated;
- NoRule: nenhuma regra precisa ser verificada.

E.1.9 ERP

É o principal componente do metamodelo que possui todas as entidades associadas assim como os tipos customizados. Ao criar o modelo é esse componente que deve ser selecionado para ser "Model Object".

E.2 Sobre a relação entre componentes

Componentes relacionados possuem regras e comportamentos que garantem que a regra de negócio seja respeitada. O comportamento (behavior) define o que deve ser feito ao criar ou remover um registro, por exemplo, ao criar uma compra no sistema o atributo relacionado quantidade da entidade relacionada ao produto deve ser somado ao atributo quantidade do produto. Em outras palavras, ao gerar uma compra a entrada no estoque é feita através do BehaviorType Add, ao excluir a compra o estoque deve ser retirado através do BehaviorType Subtract.

Já a regra é usada para validação, por exemplo, para fazer a locação de produtos não deve ser permitido retirar mais do produto do que existe no estoque. Nesse caso a RuleType é SmallerOrEqual. Em outras palavras, a quantidade informada no momento da locação deve ser menor ou igual que existe em estoque.

E.3 Atualização

No Moodle está disponível para download o material para atualização, os arquivos nessa pasta compactada devem ser substituídos em C:(a hierarquia de pastas deve coincidir). Depois de feita a substituição o projeto no Eclipse deve ser compilado novamente: Project -> Clean -> Clean all projects -> OK.

E.4 Entrega

Durante o estudo de caso um formulário deverá ser preenchido informando os passos do desenvolvimento. Esse formulário será entregue de duas formas, escrita e digital; a primeira será preenchida ao longo do exercício, a segunda após a conclusão, ambas entregas deverão ter exatamente o mesmo conteúdo.

Após o estudo de caso outro formulário deverá ser preenchido e entregue apenas de forma digital.

Os arquivos gerados deverão ser entregues via Moodle. O modelo gerado (My.metaerp), a pasta do projeto CakePHP e o dump do banco de dados (metausaerpweb.sql), a hierarquia da pasta deverá ser:

- MetaUsaERPWeb/
  - CakePHP/
  - metausaerpweb.sql
  - My.metaerp

A pasta deverá ser compactada (zip ou rar) e enviada.

Grupos que não enviarem todos os arquivos e/ou não enviarem os dois formulários caracterizarão entrega incompleta.

Apêndice F

Formulário de Coleta de Dados - Tradicional

Grupo nº		
	RA	**Nome**
Integrante 1		
Integrante 2		
Integrante 3		
Integrante 4		

Anote na tabela abaixo os horários de início e fim de cada atividade realizada.

Atividade	**Hora de início**	**Hora de término**
Projetar (modelagem, decisões de tecnologia, padrões, frameworks, etc)	____:____h	____:____h
Implementar	____:____h	____:____h
Testar	____:____h	____:____h

Anote as quantidades de linha de código geradas automaticamente e manualmente. Considere como linhas de código geradas automaticamente todo código que é criado automaticamente pelo IDE (exemplo: templates de páginas Web, templates de CSS, etc) ou por qualquer mecanismo de geração automática de código que possa ter sido empregado pelo grupo.

Inclua na contagem apenas as linhas de código das páginas Web, arquivos JavaScript e arquivos CSS customizados que não fazem parte de bibliotecas reutilizadas. Não inclua, por exemplo, arquivos da biblioteca jQuery e CSS de temas do jQuery, dentre outros.

Linhas de código do Model geradas **automaticamente** pelas transformações	
Linhas de código do Model geradas **manualmente**	
Linhas de código do Controller geradas **automaticamente** pelas transformações	
Linhas de código do Controller geradas **manualmente**	
Linhas de código da View geradas **automaticamente** pelas transformações	
Linhas de código da View geradas **manualmente**	
Linhas totais de código geradas **automaticamente** pelas transformações	
Linhas totais de código geradas **manualmente**	

Anote os problemas técnicos encontrados durante a execução do experimento, indicando o horário de identificação e resolução do problema, bem como sua breve descrição.

Descrição do problema	Hora de identificação	Hora de resolução
	____:____h	____:____h
	____:____h	____:____h
	____:____h	____:____h
	____:____h	____:____h
	____:____h	____:____h

Apêndice G

Formulário de Coleta de Dados - MetaUsaERPWeb

Grupo nº		
	RA	**Nome**
Integrante 1		
Integrante 2		
Integrante 3		
Integrante 4		

Anote na tabela abaixo os horários de início e fim de cada atividade realizada.

Atividade	**Hora de início**	**Hora de término**
Projetar (Modelagem)	____:____h	____:____h
Implementar	____:____h	____:____h
Testar	____:____h	____:____h

Anote as quantidades de linha de código geradas automaticamente e manualmente. Para a contagem do código gerado de forma automática, efetue a contagem logo após a aplicação das transformações Modelo para Código. Em seguida, complemente a implementação se necessário. Realize novamente a contagem. A diferença entre a contagem atual e a contagem anterior será a quantidade de linhas de código implementadas manualmente.

Inclua na contagem apenas as linhas de código das páginas Web, arquivos JavaScript e arquivos CSS customizados que não fazem parte de bibliotecas reutilizadas. Não inclua, por exemplo, arquivos da biblioteca jQuery e CSS de temas do jQuery, dentre outros.

Linhas de código do Model geradas **automaticamente** pelas transformações	
Linhas de código do Model geradas **manualmente**	
Linhas de código do Controller geradas **automaticamente** pelas transformações	
Linhas de código do Controller geradas **manualmente**	
Linhas de código da View geradas **automaticamente** pelas transformações	
Linhas de código da View geradas **manualmente**	
Linhas totais de código geradas **automaticamente** pelas transformações	
Linhas totais de código geradas **manualmente**	

Anote os problemas técnicos encontrados durante a execução do experimento, indicando o horário de identificação e resolução do problema, bem como sua breve descrição.

Descrição do problema	Hora de identificação	Hora de resolução
	____:____h	____:____h
	____:____h	____:____h
	____:____h	____:____h
	____:____h	____:____h
	____:____h	____:____h

Apêndice H

Formulário de Avaliação - Tradicional

Itens marcados com um asterisco (*) são de preenchimento obrigatório.

1. **Grupo** *
   Marcar apenas uma oval.
   - ◯ Grupo 1
   - ◯ Grupo 3
   - ◯ Grupo 5
   - ◯ Grupo 7
   - ◯ Grupo 9
   - ◯ Grupo 11
   - ◯ Grupo 13

2. **RA** *

   _____

3. **Você utilizou algum mecanismo e/ou ferramenta para suporte à geração automática do módulo ERP? Esse mecanismo/ferramenta atendeu satisfatoriamente suas necessidades?** *

   _____

   _____

   _____

   _____

   _____

4. **Você sentiu dificuldades na realização das atividades para o desenvolvimento do módulo ERP?** *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Parcialmente
- ◯ Não

5. **Especifique sua resposta anterior** *

_____

_____

_____

_____

_____

6. **Você considera que um processo que forneça um roteiro e mecanismos de apoio especificamente voltados para o desenvolvimento de módulo ERP facilitaria sua tarefa?** *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Parcialmente
- ◯ Não

7. **Justifique a resposta anterior** *

_____

_____

_____

_____

_____

8. **Observações que desejo fazer**

_____

_____

_____

_____

_____

Apêndice I

Formulário de Avaliação - MetaUsaERPWeb

Itens marcados com um asterisco (*) são de preenchimento obrigatório.

1. **Grupo** *
   Marcar apenas uma oval.

   - ◯ Grupo 2
   - ◯ Grupo 4
   - ◯ Grupo 6
   - ◯ Grupo 8
   - ◯ Grupo 10
   - ◯ Grupo 12
   - ◯ Grupo 14

2. **RA** *

   _____

3. **Você considera que a aplicação do MetaERPWeb auxiliou no desenvolvimento do módulo ERP?** *
   Considere o metamodelo e as transformações
   Marcar apenas uma oval.

   - ◯ Sim
   - ◯ Parcialmente
   - ◯ Não

4. **Justifique a resposta anterior** *

5. **Você sentiu dificuldades na realização das atividades?** *
Marcar apenas uma oval.
- Sim
- Parcialmente
- Não

6. **Especifique sua resposta anterior** *

7. **Quais sugestões você teria para melhorar o processo?** *

8. **Você acredita que o uso da linguagem específica de domínio usada para a modelagem do módulo ERP facilitou o desenvolvimento da aplicação?** *
Marcar apenas uma oval.
- Sim
- Parcialmente
- Não

9. **Justifique o item anterior** *

10. **Você considera viável, sob o ponto de vista de esforço de desenvolvimento, a adoção do MetaERPWeb para aplicações ERP complexas?** *
Considere que o metamodelo gera o código parcial que terá que ser completado de forma manual
Marcar apenas uma oval.

- ◯ Sim
- ◯ Parcialmente
- ◯ Não

11. **Justifique o item anterior** *

12. **Observações que desejo fazer**

Apêndice J

Pré-Questionário para Avaliação Heurística

1. **Sua idade** *
   Marcar apenas uma oval.
   - ◯ De 18 a 24 anos
   - ◯ De 25 a 30 anos
   - ◯ De 30 a 40 anos
   - ◯ Mais que 40 anos

2. **Aluno de** *
   Marcar apenas uma oval.
   - ◯ Graduação
   - ◯ Mestrado Regular
   - ◯ Mestrado Especial

3. **Ano de ingresso** *

   _____

4. **Mês/Ano de conclusão** *
   Ou previsão de conclusão - No "Dia" insira 01

   _____
   Exemplo: 15 de dezembro de 2012

5. **A - Em relação à apresentação visual de interfaces interativas marque o grau de importância que você atribui para cada um dos itens abaixo:** *

Marcar apenas uma oval por linha.

	Nada importante	Pouco importante	Importante	Significativamente importante	Muito importante
1. Identificar visualmente a sua localização no sistema para saber onde está e para onde pode ir.	◯	◯	◯	◯	◯
2. O conteúdo da tela distribuído e posicionado seguindo uma ordem natural e lógica.	◯	◯	◯	◯	◯
3. Tela não contendo informação irrelevante ou raramente necessária.	◯	◯	◯	◯	◯
4. Indicação do formato correto de entrada de dados (data, CEP, telefone, etc.).	◯	◯	◯	◯	◯
5. Destaque nos campos de formulário que são de preenchimento obrigatório.	◯	◯	◯	◯	◯
6. Palavras e termos adequados ao contexto da aplicação que podem ser facilmente lembrados.	◯	◯	◯	◯	◯

6. **B - Em relação às mensagens de feedback do sistema marque o grau de importância que você atribui para cada um dos itens abaixo:** *

Marcar apenas uma oval por linha.

	Nada importante	Pouco importante	Importante	Significativamente importante	Muito importante
1. Mensagens informativas sobre o que está acontecendo no sistema durante a interação.	⬭	⬭	⬭	⬭	⬭
2. Uso de linguagem familiar que pode ser facilmente interpretada.	⬭	⬭	⬭	⬭	⬭
3. Padronização das mensagens, formatos, símbolos e cores do sistema.	⬭	⬭	⬭	⬭	⬭
4. Alerta sobre formatos incorretos ou dados inconsistentes.	⬭	⬭	⬭	⬭	⬭
5. Mensagens de erro visíveis, simples e de fácil entendimento.	⬭	⬭	⬭	⬭	⬭

7. **C - Quanto à navegabilidade em sistemas interativos marque o grau de importância que você atribui para cada um dos itens a seguir:** *

Marcar apenas uma oval por linha.

	Nada importante	Pouco importante	Importante	Significativamente importante	Muito importante
1. Apoio para fazer e desfazer ações.	⬭	⬭	⬭	⬭	⬭
2. Controle de localização para voltar ao ponto de partida ou sair de um estado inesperado.	⬭	⬭	⬭	⬭	⬭
3. Mensagens que previnem a ocorrência de problemas no caso de ações equivocadas.	⬭	⬭	⬭	⬭	⬭
4. Os botões de ação conforme identificados definem claramente o estado que será atingido após seu acionamento.	⬭	⬭	⬭	⬭	⬭
5. Informação organizada em ordem alfabética ou lógica por relevância de uso.	⬭	⬭	⬭	⬭	⬭
6. Elementos de interface dispostos de forma a minimizar o esforço de ações físicas e nas buscas visuais.	⬭	⬭	⬭	⬭	⬭

8. **D - A respeito do suporte a tarefas em sistemas interativos marque o grau de importância que você atribui para cada um dos itens abaixo:** *

Marcar apenas uma oval por linha.

	Nada importante	Pouco importante	Importante	Significativamente importante	Muito importante
1. Apoio para fazer e desfazer ações.	◯	◯	◯	◯	◯
2. O sistema fornece uma sequência de operações para completar tarefas.	◯	◯	◯	◯	◯
3. Automatização de tarefas rotineiras e redundantes.	◯	◯	◯	◯	◯
4. Filtros de busca adequados ao número de itens e informações.	◯	◯	◯	◯	◯
5. Mensagens que auxiliam na recuperação de erros e mostra como acessar soluções alternativas.	◯	◯	◯	◯	◯
6. Apoio às tarefas complexas com instruções visíveis e acessíveis.	◯	◯	◯	◯	◯

9. **Experiência com o conceito de Enterprise Resource Planning (ERP)** *

Selecione a opção que melhor reflita seu grau atual de experiência.

Marcar apenas uma oval.

◯ Eu não tenho familiaridade com este assunto. Eu nunca fiz isto.

◯ Li ou estudei sobre isto. Conheço os conceitos e/ou técnicas.

◯ Eu utilizo isto algumas vezes em projetos pessoais ou na indústria, mas não sou um(a) especialista.

◯ Eu sou muito familiar com este assunto. Eu me sentiria confortável fazendo isto.

10. **Experiência com a técnica de Avaliação Heurística** *

Selecione a opção que melhor reflita seu grau atual de experiência.

Marcar apenas uma oval.

◯ Eu não tenho familiaridade com este assunto. Eu nunca fiz isto.

◯ Li ou estudei sobre isto. Conheço os conceitos e/ou técnicas.

◯ Eu utilizo isto algumas vezes em projetos pessoais ou na indústria, mas não sou um(a) especialista.

◯ Eu sou muito familiar com este assunto. Eu me sentiria confortável fazendo isto.

11. **Experiência e formação na área de Computação** *
Selecione todas as opções que reflitam sua experiência e formação (pode selecionar mais de uma).
Marque todas que se aplicam.

- [ ] Sou graduado na área de Computação.
- [ ] Trabalho na área de desenvolvimento de software.
- [ ] Realizo estágio na área de desenvolvimento de software.
- [ ] Sou docente de cursos de Computação.
- [ ] Sou apenas estudante.

12. **Indique o tempo de experiência na área de Computação**
Responda somente se você não é apenas estudante.
Marcar apenas uma oval.

- ◯ Menos de 3 anos.
- ◯ De 3 a 5 anos.
- ◯ De 6 a 10 anos.
- ◯ Mais de 10 anos.

Apêndice K

Formulário de Tarefas para Avaliação Heurística

Orientações:

1. Preencher dados gerais da inspeção: horário de início e fim e nome do aluno
2. Você deverá inspecionar todas heurísticas nas funcionalidades apontadas nesta inspeção (tarefas descritas a seguir).
3. Descrever o problema, apontar as heurísticas violadas e sinalizar a(s) tarefa(s) (T1, T2, T3, T4, T5, T6, T7, T8) onde foi encontrada a violação.

Tarefas:

1. Cadastre o produto "Blu-ray Trilogia O Senhor dos Anéis"ao valor de R$ 199,90, 100 unidades em estoque e descrição "Conheça mais profundamente o universo de Tolkien"
2. Faça a venda no dia 21/08/2014 de 50 unidades do produto "Blu-ray Trilogia O Senhor dos Anéis"ao valor de R$ 189,90 cada
3. Informe que o valor do produto "Blu-ray Trilogia O Senhor dos Anéis"teve um desconto de R$ 5,00
4. Faça a venda no dia 28/08/2014 de 90 unidades do produto "Blu-ray Trilogia O Senhor dos Anéis"ao valor padrão
5. Cadastre um novo produto, "Blu-Ray - House M. D. - A Série Completa", ao valor de R$ 799,90 e não informe estoque nem descrição
6. Faça a correção da venda do dia 21/08/2014, alterando a quantidade vendida para 45 unidades
7. Cancele a venda do dia 28/08/2014
8. Faça a venda no dia 04/09/2014 de -2 unidades do produto "Blu-ray Trilogia O Senhor dos Anéis"ao valor de R$ -189,90 cada

Inspeção de Usabilidade				
Horário de início da inspeção:	1º dia	00h00	Nome do aluno	
Horário de fim da inspeção:	1º dia	00h00	Aplicação	

	Descrição do Problema	Heurística Violada	Tarefas							
			T1	T2	T3	T4	T5	T6	T7	T8
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										